DISSERTATION

COMPARING SETS OF DATA SETS ON THE GRASSMANN AND FLAG MANIFOLDS WITH APPLICATIONS TO DATA ANALYSIS IN HIGH AND LOW DIMENSIONS

Submitted by

Xiaofeng Ma

Department of Mathematics

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2020

Doctoral Committee:

Advisor: Michael Kirby
Co-Advisor: Chris Peterson

Edwin Chong
Louis Scharf
Clayton Shonkwiler

ABSTRACT

COMPARING SETS OF DATA SETS ON THE GRASSMANN AND FLAG MANIFOLDS WITH APPLICATIONS TO DATA ANALYSIS IN HIGH AND LOW DIMENSIONS

This dissertation develops numerical algorithms for comparing sets of data sets utilizing shape and orientation of data clouds. Two key components for "comparing" are the distance measure between data sets and correspondingly the geodesic path in between. Both components will play a core role which connects two parts of this dissertation, namely data analysis on the Grassmann manifold and flag manifold.

For the first part, we build on the well known geometric framework for analyzing and optimizing over data on the Grassmann manifold. To be specific, we extend the classical self-organizing mappings to the Grassamann manifold to visualize sets of high dimensional data sets in 2D space. We also propose an optimization problem on the Grassmannian to recover missing data.

In the second part, we extend the geometric framework to the flag manifold to encode the variability of nested subspaces. There we propose a numerical algorithm for computing a geodesic path and distance between nested subspaces. We also prove theorems to show how to reduce the dimension of the algorithm for practical computations. The approach is shown to have advantages for analyzing data when the number of data points is larger than the number of features.

ACKNOWLEDGEMENTS

I have encountered many supporters for the past few years of my mathematical adventure. I am very certain that without the help and encouragement of these people, by no means I could have proceeded or persisted in this journey. Thank you for all the help!

I don't know if I can express my deep gratitude to my advisors Michael Kirby and Chris Peterson enough, for their inspiration, patience, and careful guidance. I look up to them and they mean too much for me. I want to thank Henry Adams and Louis Scharf for the inspiring conversations that we have and for the fruitful and rewarding collaborations that guide me through my expedition. I want to thank my best friend Mmanu Chaturvedi for the meaningful conversations about philosophy and life. I want to thank my parents Huimin Fu and Youwen Ma for their love and support, and I want to thank my girlfriend Siqin Huang, we met at the lowest point of my life, since when her continuous encouragement has motivated me until today.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

1.1 From comparing sets of data points to comparing sets of data sets

In this dissertation, we focus on developing algorithms for recognizing patterns within a collection of high dimensional data sets. An emphasis is on developing geometrically inspired algorithms on parameter manifolds that help make patterns within the collection of data sets more apparent. In many real world applications, we use algorithms, supervised or unsupervised, for classifying, clustering and visualizing data. Many algorithms require, for tackling such tasks, a distance measure between data points or between data sets. Examples include k-nearest neighbor, Laplacian eigenmaps, and multi-dimensional scaling. Other algorithms require further information, such as knowing how to determine, and move along a geodesic path between two points. Examples include k-means, gradient descent, and self-organizing mappings. In Euclidean space $\mathbb{R}^n$, the standard distance measure between points is the Euclidean distance and the standard geodesic connecting two points is a line segment. There are applications where a different distance metric is preferred, for example, the cosine of the angle between attributes vectors of two songs is often used in song recommendation algorithms. Using the cosine of the angle between vectors, in some sense, can be understood through the lens of data embedded on a unit $n - sphere$ (where the ambient dimension is $n+1$). The angle between two points, viewed as unit vectors, tells us how "close" the two points are following a geodesic path on the sphere. To move one point to another we simply follow the path along a great circle passing through the points as great circles are the geodesic paths on the sphere.

One can generalize this idea of comparing vectors on a sphere. In many applications, it is not a single vector itself that matters, but rather the subspace spanned by a collection of vectors in $\mathbb{R}^n$ or else a subspace minimizing some comparison function to the collection. From this vantage point, one may represent a data point/data set as a linear subspace. In this context, the natural place to perform data analysis comparisons is on a Grassmann manifold. We let $Gr(k,n)$, denote the Grassmann manifold whose points parametrizes k-dimensional linear subspaces in $\mathbb{R}^n$. The utilization of the Grassmann manifold has become widespread in computer vision and other disciplines in pattern recognition. Some examples include video processing [24], classification, [9, 23, 48, 49], action recognition [4], expression analysis [36, 44, 45], domain adaptation [33, 40], and regression [25, 43].

On the Grassmannian, two data sets, represented as two points, can be compared by computing the distance between the points. A geodesic path between two points on a Grassmannian corresponds to a minimal alignment of two linear subspaces. Both the distance and the geodesic path between points have analytical expressions thanks to the underlying geometric structure of Grassmann manifolds.

A natural generalization of the Grassmann manifold is the flag manifold. A flag manifold parametrizes sequences of nested subspaces satisfying a common length and dimension constraint. As such, flag manifolds can be understood as a generalization of Grassmann manifolds. In this dissertation, we attempt to provide effective and efficient numerical algorithms to compute the distance and geodesic path between two flags. We also prove a key theoretical result that allows one to reduce the dimension of the algorithm which increases the range of practical applicability.

The flag manifold approach being proposed here generalizes ideas related to the Grassmann manifold and can be viewed as a tool for the analysis of relationships between tall matrices while also being useful for determining relationships between classes of wide matrices. Note in each case here the data matrix columns are samples of data. We argue that the distances measured between large sets of small feature spaces captures more fidelity than algorithms on Euclidean space.

1.2 Overview

In Chapter 2, we provide some background for the well studied Grassmann manifold. Towards the purpose of comparing data sets, we introduce the necessary geometric ingredients on the Grassmannian including basic definitions, quotient structure, tangent space, geodesic path and geodesic distance. We present an explicit derivation to show the connection between geodesic paths on the Grassmannian and principal vectors between two linear subspaces. From there we can obtain a pictorial explanation of the geodesic path to better understand the algebraic formula

In Chapter 3, inspired by the classical Proper Orthogonal Decomposition (POD) algorithm, we pose an optimization problem on the Grassmannian to solve the missing data problem from a geometric prospective. We show that solving a particular optimization problem, using a steepest descent algorithm with penalty term, can effectively avoid some local minima where the POD algorithm gets stuck. This is illustrated by applying both algorithms to traveling sinusoidal wave data.

In Chapter 4, we extend the classical self-organizing mappings to the Grassmann manifold using the formula introduced in Chapter 2. We demonstrate the Grassmannian SOM by visualizing high dimensional hyper-spectral image data and gene expression data on a 2D grid. We obtain strong clustering results while Euclidean distance versions fail to obtain clear separation.

In Chapter 5, we derive some of the geometric features like tangent space, exponential and logarithmic map on the flag manifold which can be viewed as a generalization or refined version of such formulas on the Grassmannian . More importantly, we propose a numerical algorithm for computing the log map between two flags. Here we also prove theorems to run the computation in lower dimensional space. The algorithm is implemented on the Indian Pines data set to compare with Grassmannian distance.

In Chapter 6, we review our contributions and talk about some open problems where the techniques discussed in this thesis can be applied.

Chapter 2

The Grassmann Manifold

2.1 Linear Subspace Models

High-dimensional data has been well-studied in a wide range of domains such as pattern recognition, image analysis, data visualization, etc. Sometimes high-dimensional data is referred to by the term "big data". Unfortunately, this term has the potential to be ambiguous. Big data can be used to describe biological data sets consisting of gene expressions where the number of attributes (genes) is large but the number of observations is small. It can also be applied to datasets such as MNIST database which consists of 60,000 training images and 10,000 test images and each image is a handwritten digit represented by a 28×28 matrix. In this case the ambient dimension is relatively small while the number of samples is large. Datasets can be "big" in different aspects.

For clarification, in this dissertation, we will focus on the setting of high-dimensional ambient space such as Euclidean space. It is convenient to view a data point as a vector living in the n-dimensional Euclidean space $\mathbb{R}^n$ where n is the number of measured attributes. For example, we can represent a facial image by stacking the columns of an image and the resulting vector can be viewed as a data point in $\mathbb{R}^n$ where n is the number of pixels in this image. The advantage of putting an image in Euclidean space is that one may utilize various optimization algorithms, mathematical and statistical analysis tools developed in this setting.

In many applications, it is not a single vector or a set of vectors in a neighborhood in $\mathbb{R}^n$ that matters, but rather the subspace spanned by this set of vectors. For example, let $V = [v_1, v_2, \ldots, v_k]$ be a collection of images of the same subject under different illumination conditions [7, 10–12], all v_i's are distinct vectors in $\mathbb{R}^n$ with the same identity. But how is this identity to be represented while including the illumination space? This is the case where the range of V counts instead of V itself. Unless otherwise specified, we denote the subspace spanned by V as boldface $\mathbf{V}$. This pattern set framework allows us to encode the identity while capturing the variation in our model. It has been observed that by utilizing the linear subspace model, i.e. considering a set of data points as a pattern, one can improve the robustness of pattern recognition algorithms. Take illumination subspaces as an example, the resolution of images can be reduced without affecting the classification rate. The manifold whose points parameterize all k-dimensional subspaces in $\mathbb{R}^n$ is the Grassmann manifold $\mathrm{Gr}(k, n)$. Its use allows us to build a theoretical bridge to this abstract mathematical object where the geometry agrees with the structure of data.

2.2 The Grassmannian: A quotient manifold of O(n)

In this section, we introduce the geometry of the Grassmann manifold as well as its geodesic formula.

Definition 2.2.1. (Grassmann manifolds). Let $\mathbf{V}$ be an n-dimensional real vector space, for any $0 \leq k \leq n$ we define the Grassmannian, $\mathrm{Gr}(k, n)$, as the set of all k-dimensional subspaces of $\mathbf{V}$.

In this dissertation, we denote by $O(k)$ the orthogonal group of k-by-k matrices, $SO(k)$ denotes the special orthogonal group of k-by-k matrices with determinant 1 and $GL(k)$ denotes the linear group of k-by-k invertible matrices.

It is convenient to view the Grassmannian $Gr(k, n)$ as the quotient manifold $O(n)/O(k) \times O(n-k)$. Let $Q \in O(n)$ be an n-by-n orthogonal matrix, the equivalence class $[Q]$ is the set of all orthogonal matrices whose first k columns span the same subspace as the one spanned by the first k columns of Q. A point on the Grassmann manifold is the equivalence class,

$$[Q] = \left\{ Q \begin{pmatrix} Q_k & 0 \\ 0 & Q_{n-k} \end{pmatrix} : Q_k \in O(k),\ Q_{n-k} \in O(n-k) \right\}. \tag{2.1}$$

One advantage of this approach is we may utilize the orthogonal group geodesic and the quotient geometry of the Grassmann manifold. This quotient space representation also reveals that the dimension of $\mathrm{Gr}(k, n)$ is $\frac{n(n-1)}{2} - [\frac{k(k-1)}{2} + \frac{(n-k)(n-k-1)}{2}] = k(n-k)$, which can also be verified when we look at the tangent space at $[Q]$.

2.2.1 Tangent space of $O(n)$

A p-dimensional embedded submanifold can be approximated locally as $\mathbb{R}^p$. The tangent space to a point can be visualized as a linear space tangent to the submanifold at that point. $O(n)$ is an $n(n-1)/2$-dimensional submanifold of $\mathbb{R}^{n\times n}$, therefore the tangent space at a point on $O(n)$ is an $n(n-1)/2$ dimensional vector space centered at the point of tangency. One may derive the tangent space to $O(n)$ at a point $Q \in O(n)$, denoted by $T_Q O(n)$, as follows:

Let $\epsilon > 0$,

$$I_\epsilon = \{t \in \mathbb{R} \mid -\epsilon < t < \epsilon\}$$

and let

$$X(t) : I_\epsilon \mapsto O(n)$$

be any smooth curve on $O(n)$ such that $X(0) = Q$. Since

$$X(t)^T X(t) = I \ , t \in I_\epsilon$$

differentiating both sides yields

$$\dot{X}(t)^T X(t) + X(t)^T \dot{X}(t) = 0$$

let $t = 0$ we have

$$\dot{X}(0)^T Q + Q^T \dot{X}(0) = 0$$

This derivation shows that

$$T_QO(n) \subset \{\Delta | \Delta^T Q + Q^T \Delta = 0\}$$

By counting the dimension of $O(n)$ and Δ it can be readily verified that

$$T_QO(n) = \{\Delta | \Delta^T Q + Q^T \Delta = 0\}, \tag{2.2}$$

i.e., the tangent space to $O(n)$ at Q is the set of matrices Δ where $Q^T\Delta$ is any n-by-n skew-symmetric matrix.

2.2.2 $\mathrm{Gr}(k, n)$ as a quotient manifold of $O(n)$

The fact that $\mathrm{Gr}(k, n)$ is a quotient space of $O(n)$ provides us a way to find a concrete numerical representation for tangent vectors to this abstract mathematical manifold.

Let $Q \in O(n)$ and $[Q] \in Gr(k, n)$, one can think of a tangent vector V to $Gr(k, n)$ at $[Q]$ as a variation of the k-dimensional subspace spanned by the first k column of Q. To obtain a matrix representation of V, the key idea is to decompose $T_QO(n)$ into a component which does not modify the span and a component which modifies the span. The latter represents a tangent vector V to $Gr(k, n)$ at $[Q]$. This idea is demonstrated in Figure 2.1.

Vertical and Horizontal Space

Here we define the **vertical space** V_Q as the set of matrices which preserve the span of the first k columns of Q, i.e., the tangent space to the equivalence class $[Q]$ at Q.

The computation of the vertical space at a point $Q \in O(n)$ is essentially the same as the computation of the tangent space to $O(n)$. Let $V(t) = Q \begin{pmatrix} Q_k(t) & 0 \\ 0 & Q_{n-k}(t) \end{pmatrix}$ be any smooth curve living in the equivalence class $[Q]$ with $Q_k(0) = I_k$ and $Q_{n-k}(0) = I_{n-k}$, i.e. $V(0) = Q$. We have $V(t)^T V(t) = I$. Differentiating both sides and evaluating at $t = 0$ yields

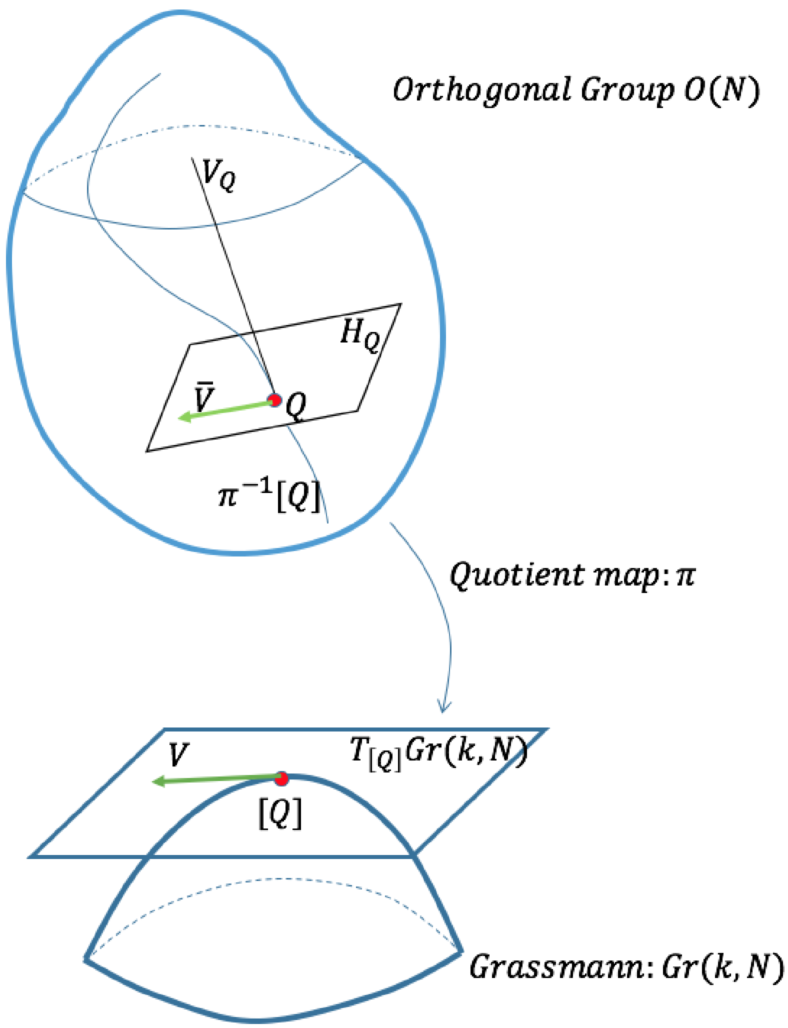


Figure 2.1: This figure illustrates the idea of the vertical space V_Q and horizontal space H_Q at a point Q. $T_{[Q]}\mathrm{Gr}(k, n)$ is the tangent space to $\mathrm{Gr}(k, n)$ at $[Q]$. It can be shown that a tangent vector $V \in T_{[Q]}\mathrm{Gr}(k, n)$ can be uniquely represented by a tangent vector $\overline{V} \in H_Q$. Hence one can represent tangent vectors to a point on $\mathrm{Gr}(k, n)$ as matrices.

$$\dot{Q}_k(0)^T + \dot{Q}_k(0) = 0$$
$$\dot{Q}_{n-k}(0)^T + \dot{Q}_{n-k}(0) = 0$$

With this computation, along with a dimension count, one may show that the vertical space at a point Q is the set of matrices

$$V_Q = \left\{ Q \begin{pmatrix} C & 0 \\ 0 & D \end{pmatrix} \right\},$$

where C is any k-by-k skew-symmetric matrix and D is any $(n-k)$-by-$(n-k)$ skew-symmetric matrix.

The **horizontal space** H_Q is defined as the orthogonal complement of V_Q in $T_Q O(n)$, which consists of the tangent vectors that modify the span of the first k columns of Q. The orthogonality is with respect to the Euclidean metric defined in Definition 2.2.2. In this thesis, we use the Euclidean metric as our Riemannian metric.

Definition 2.2.2. Let $U, V \in T_Q O(n)$, the Euclidean metric is defined as a function $e : T_Q O(n) \times T_Q O(n) \mapsto \mathbb{R}$:

$$\begin{aligned} e(U, V) &= \mathrm{Tr}(U^T V) \\ &= \mathrm{vec}(U)^T \mathrm{vec}(V) \end{aligned}$$

The horizontal space H_Q is the set of matrices which are orthogonal to the vertical space and living in $T_Q O(n)$. To compute H_Q, we need to solve the following set of equations

$$\mathrm{Tr}(\Delta^T Q \begin{pmatrix} C & 0 \\ 0 & D \end{pmatrix}) = 0$$
$$\Delta = QA$$

where $A \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{k\times k}$ and $D \in \mathbb{R}^{(n-k)\times(n-k)}$ are skew-symmetric matrices. The solution to these equations, i.e. the horizontal space at Q, is the set of matrices

$$H_Q = \left\{ Q \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \right\}.$$

Recall that $V_Q O(n)$ is the set of matrices $\{\Delta | \Delta^T Q + Q^T \Delta = 0\}$. It can be readily verified that $T_Q O(n) = V_Q \bigoplus H_Q$.

Intuitively, one can think of the vectors in the vertical space as the set of velocity vectors which preserve the equivalence class while the vectors in the horizontal space modify the equivalence class. Hence, tangent vectors to geodesics on $Gr(k,n)$ must be living in the horizontal space. As depicted in Figure 2.1, if V is a tangent vector to $Gr(k,n)$ at $[Q]$, then there is a horizontal vector $\overline{V} \in H_Q$ which represents V uniquely. The horizontal space gives a matrix representation for any tangent vector V to $Gr(k,n)$ at $[Q]$.

Geodesic Formula

The intuition behind geodesics on manifolds is analogous to the picture of straight lines in $\mathbb{R}^n$. Geometrically, a straight line in R^n can be thought of as a parametrized curve $\gamma(t) : [0,1] \mapsto \mathbb{R}^n$ without acceleration for all $t \in [0,1]$. Additionally, a minimal geodesic between two points represents the shortest curve connecting two points. The associated length function which will be discussed more in section 2.4.3. In [19], the authors show the geodesic formula on $O(n)$ is given by $Q(t) = Q\exp(tA)$ where A is an n-by-n skew-symmetric matrix.

$$Q(t) = Q \exp \left(t \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \right)$$

has a horizontal tangent vector

$$\dot{Q}(t) = Q(t)\begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix}$$

for all t along $Q(t)$. Therefore $Q(t)$ is still the shortest path on the quotient space $Gr(k, n)$, i.e. by further restricting A to be of the form

$$\tilde{A} = \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix}, B \in \mathbb{R}^{(n-k)\times(k)}$$

we obtain a representative of the geodesic path on $Gr(k, n)$

$$Q(t) = Q\exp(t\tilde{A}).$$

The $(n-k)$-by-(k) submatrix B specifies the velocity of the geodesic flow. This approach provides us an easy method to compute the geodesic formula on the Grassmann manifold using n-by-k matrices.

2.3 The Grassmannian: A quotient manifold of the Stiefel

In practical applications, we are usually interested in the span of the first k columns of $Q(t)$, hence the geodesic formula can be rewritten as,

$$\Phi(t) = Q\exp(t\tilde{A})J$$

where $J = \begin{bmatrix} I_k \\ 0_{n-k,k} \end{bmatrix} \in \mathcal{R}^{n\times k}$ and $\Phi(t)$ is a n-by-k orthonormal matrix for any $t \in [0, 1]$.

Here we introduce the **Stiefel manifold** as the set of all orthonormal k-frames in $\mathbb{R}^n$, denoted by $St(k, n)$. Each point on $St(k, n)$ can be expressed as an n-by-k orthonormal matrix Y such that $Y^TY = I_k$. In this way, the Grassmann manifold $Gr(k, n)$ can be viewed as a quotient manifold of the Stiefel Manifold $St(k, n)$. An equivalence class $[Y]$ can be defined as

$$[Y] = \{YD : D \in O(k)\}.$$

It is straightforward to show the tangent space to $St(k,n)$ at Y, $T_Y St(k,n)$, is,

$$\{\Delta | Y^T \Delta + \Delta^T Y = 0\}$$

Let $Y \in St(k,n)$ and $[Y] \in Gr(k,n)$, by utilizing the quotient geometry of $Gr(k,n)$, we can compute the corresponding vertical space V_Y and horizontal space H_Y,

$$V_Y = \{YA : A \in \mathbb{R}^{k \times k}, A^T + A = 0\}.$$

And the horizontal space is

$$H_Y = \{N : N \in \mathbb{R}^{n \times k}, N^T Y = 0\}.$$

A tangent vector to the Grassmann manifold at $[Y]$ can be uniquely represented by an n-by-k matrix N where $Y^T N = 0$.

In [19], Edelman $et\ al$ present the following theorem to compute the geodesic flow on $Gr(k,n)$.

Theorem 1. *If* $\Phi(t) = Q \exp(t\tilde{A})J$*, with* $\Phi(0) = X$ *and* $\dot{\Phi}(0) = H$ *where* $\tilde{A} = \begin{bmatrix} 0 & -B^T \\ B & 0 \end{bmatrix}$*, then*

$$\Phi(t) = XV\cos(\Sigma t)V^T + U\sin(\Sigma t)V^T \tag{2.3}$$

where $U\Sigma V^T$ *is the compact singular value decomposition (SVD) of* H*. Here* U *is an* n*-by-*k *orthonormal matrix,* Σ *is a* k*-by-*k *diagonal matrix and* V *is a* k*-by-*k *orthogonal matrix.*

The proof of this Theorem is given in [19]. Please note that $\phi(t)$ is no more than a representative of an equivalence class $[\phi(t)]$ because the Grassmann manifold is a quotient manifold of the Stiefel

Algorithm 1: Geodesic curve starting from $[X]$ in the direction H

Input Data: Initial Position:$X \in \mathbb{R}^{n \times k}$, $X^T X = I$; Initial Velocity:$H \in$ Horizontal Space; time: $t \in \mathbb{R}$

Output Data: $\phi(t) \in St(k, n)$

```
Function Geodesic(X, V, t):
    UΣV^T = thin SVD(H);
    return X V cos(Σt) V^T + U sin(Σt) V^T;
```

manifold. Pseudocode for tracing a geodesic curve in the direction of a given tangent vector can be found in Algorithm 1.

2.4 Geodesic formula: Exponential and Logarithmic map

2.4.1 Exponential map

The geodesic formula introduced above solves the following problem: given initial conditions, i.e. an initial position X and an initial velocity H, find the resulting geodesic on the Grassmann manifold. This process can be thought of as mapping tangent vectors back onto $Gr(k, n)$, which is closely related to the idea of the **exponential map**.

Definition 2.4.1. For every $H \in T_{[X]}Gr(k, n)$ there is a unique geodesic $\phi(t)$: $[0, 1] \mapsto Gr(k, n)$ such that $\phi(0) = X$ and $\phi'(0) = H$. The mapping $\mathrm{Exp}_X : T_X Gr(k, n) \mapsto Gr(k, n)$,

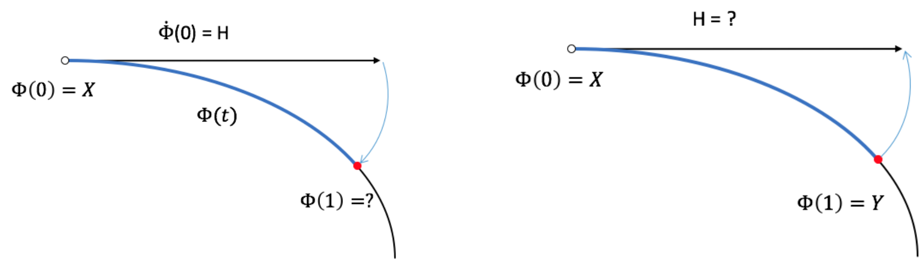
$$\mathrm{Exp}_X(H) \doteq \phi(1)$$

is called the **exponential map** of H at [X].

An illustration of the exponential map is shown in Figure 2.2(a). On $Gr(k, n)$, the exponential map can be computed via Algorithm 1 by letting $t = 1$.

2.4.2 Logarithmic map

In many applications, one might be interested in the inverse operation: Given n-by-k orthonormal matrices X and Y representing equivalence classes $[X]$ and $[Y]$ on $Gr(k, n)$, find an appropriate velocity matrix H such that a geodesic with velocity H, starting at $[X]$, reaches $[Y]$ in unit time. The idea behind the inverse operation is illustrated in Figure 2.2(b).



(a) Given initial position and velocity, find geodesic flow. The formula is given in equation (2.3).

(b) Given two points on Grassmann, recover the velocity which induces a geodesic flow between points. The formula is given in equation (2.6).

Figure 2.2: This figure illustrates the two problems related to the geodesic formula

Instead of computing H directly, we assemble H via its compact SVD $H = M\Theta V^T$. By Theorem 1, at $t = 1$, we have

$$YD = XV\cos(\Theta)V^T + M\sin(\Theta)V^T$$

where D is any k-by-k orthogonal matrix (since we are only required to reach a point in the equivalence class, i.e. $YD \in [Y]$). H is in the tangent space, hence it can be readily verified that $X^T H = 0$ and consequently $X^T U = 0$. Multiplying by X^T on both sides of the equation yields

$$V\cos(\Theta)V^T = X^T YD \tag{2.4}$$

$$M\sin(\Theta)V^T = (I - XX^T)YD \tag{2.5}$$

Then,

$$
\begin{aligned}
U\sin(\Theta)V^T(V\cos(\Theta)V^T)^{-1} &= Utan(\Theta)V^T \\
&= (I - XX^T)YD(X^TYD)^{-1} \\
&= (I - XX^T)Y(X^TY)^{-1}
\end{aligned}
$$

Therefore, to find the velocity matrix H, it suffices to compute the compact SVD,

$$(I - XX^T)Y(X^TY)^{-1} = U\Sigma V^T,$$

and $H = U\Theta V^T$ where $\Theta = \arctan(\Sigma)$. One subtlety in Equation (2.3) is that if V is multiplied from the right on both sides of the equation, we still have a representative of the same equivalence class as $\Phi(t)$, i.e. $\Phi(t)V$ is equivalent to $\Phi(t)$ on $Gr(k, n)$ for all t. To summarize the derivation above, we present the formula for computing the geodesic path between two points $X, Y \in Gr(k, n)$, which can be found in [1].

$$G(t) = XV\cos\Theta t + U\sin\Theta t \tag{2.6}$$

We observe that

$$[G(0)] = [X]$$

and

$$[G(1)] = [Y]$$

and the trajectory $G(t)$ traces out the path of shortest distance on $Gr(k, n)$ in terms of the geodesic metric given by Equation (2.8). The quantities U, Σ and V are found by computing the singular value decomposition of the projection of

$$M = Y(X^TY)^{-1}$$

onto the orthogonal complement of X, i.e.,

$$U\Sigma V^T = (I - XX^T)Y(X^TY)^{-1}$$

where X and Y are given and the inverse of X^TY exists. Further, it can be shown that

$$\Theta = \text{atan}(\Sigma)$$

to complete the requirements of computing the geodesic between two subspaces X and Y as prescribed in Equation (2.6). This formula is a key ingredient for extending the self-organizing mapping algorithm on vector spaces to Grassmannians.

We present the following example as an illustration of the numerical computation of the geodesic formula between two points.

Let $X = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$ and $Y = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} \\ 0 & \frac{1}{\sqrt{3}} \\ 0 & \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{2}} & 0 \end{pmatrix}$ be two matrices representing points $[X]$ and $[Y]$ on $Gr(2,4)$. First, we compute the compact singular value decomposition

$$U\Sigma V^T = (I - XX^T)Y(X^TY)^{-1}.$$

We find

$$\Sigma = \begin{pmatrix} 1.6180 & 0 \\ 0 & 0.6180 \end{pmatrix}.$$

Compute arctangent along the diagonal we get

$$\Theta = \begin{pmatrix} 1.0172 & 0 \\ 0 & 0.5536 \end{pmatrix},$$

hence

$$\cos \Theta t = \begin{pmatrix} \cos(1.0172t) & 0 \\ 0 & \cos(0.5536t) \end{pmatrix}$$

and

$$\sin \Theta t = \begin{pmatrix} \sin(1.0172t) & 0 \\ 0 & \sin(0.5536t) \end{pmatrix}.$$

Now we can sample points along this geodesic.

For $t = 0$,

$$G(0) = \begin{pmatrix} -0.5257 & -0.8507 \\ 0.8508 & -0.5257 \\ 0 & 0 \\ 0 & 0 \end{pmatrix},$$

whose column vectors span the same subspace as the column vectors of X. The same can also be verified for Y and $G(1)$ by computing the principal angles between Y and $G(1)$. If t is sampled uniformly on the interval $[0, 1]$, one can verify that all the distances between any pair of adjacent points are the same. i.e. the geodesic has a constant speed. One example is at $t = \frac{1}{2}$, the geodesic distance between X and $G(\frac{1}{2})$ is the same as the distance between $G(\frac{1}{2})$ and Y. i.e. $d_g(X, G(\frac{1}{2})) = d_g(G(\frac{1}{2}), Y)$.

2.4.3 Principal angles and distance metric

We provide a summary of the computation of angles between subspaces initially described in [8]. Let X and Y be two vector subspaces of $\mathbb{R}^n$ such that

$$p = \dim(X) \geq \dim(Y) = q \geq 1,$$

then the *principal angles* $\theta_k \in [0, \frac{\pi}{2}], 1 \leq k \leq q$ between X and Y are defined recursively by

$$\cos(\theta_k) = \max_{u \in X} \max_{v \in Y} u^T v = u_k^T v_k, \quad k = 1 \dots q \tag{2.7}$$

subject to $||u|| = ||v|| = 1, u^T u_i = 0$ and $v^T v_i = 0$ for $i = 1 \dots k-1$. Clearly, the principal angles satisfy $0 \leq \theta_1 \leq \theta_2 \leq \dots \theta_q \leq \frac{\pi}{2}$. Henceforth, $\theta = (\theta_1, \dots, \theta_q)$ will denote the principal angle vector. Note that we have abused notation somewhat in using X to represent both a subspace and an orthonormal matrix whose columns span this space. For additional details related to algorithms for the computation of principal angles, please see [8].

Let A, B be two points on the Grassmannian $Gr(k, n)$. Again, we are thinking of these points as subspaces though they are represented by orthonormal matrices whose columns span the subspaces. The geodesic distance between these two points is given by

$$d_g(A, B) = \|(\theta_1, \dots, \theta_k)\|_2 \tag{2.8}$$

Other metrics are possible, e.g., the chordal distance

$$d_g(A, B) = \|(\sin(\theta_1), \dots, \sin(\theta_k))\|_2 \tag{2.9}$$

We note that it is possible to show that the Grassmannian may be isometrically embedded into the Euclidean space when the chordal metric is employed and this is not the case for the geodesic metric [16]; see also [14].

Principal angles between subspaces are defined regardless of the dimensions of the subspaces, which are denoted, e.g., $\dim A$. Thus, inspired by the Riemannian geometry of the Grassmannian, we may define, for any vector subspaces A, B of $\mathbb{R}^n$ the *geodesic* distance

$$d_g(A, B) = \|(\theta_1, \dots, \theta_\ell)\|_2,$$

for any $\ell \leq \min\{\dim A, \dim B\}$. While d_ℓ is not, strictly speaking, a metric (for example, if $\dim A \cap B \geq \ell$, then $d_\ell(A, B) = 0$), it nevertheless provides an efficient and useful tool for analyzing configurations in $\cup_{k \geq \ell} Gr(k, n)$. The geometry driving these distance measures is captured by the notion of a Schubert variety $\bar{\Omega}_\ell(W) \subseteq Gr(k, n)$. Let W be a subspace of $\mathbb{R}^n$, then we define

$$\bar{\Omega}_\ell(W) = \{E \in Gr(k, n) \,|\, \dim(E \cap W) \geq \ell\}.$$

With this notation, $d_\ell(A, B)$ simply measures the distance between A and $\bar{\Omega}_\ell(B)$, i.e. $d(A, \bar{\Omega}_\ell(B)) = min\{d_k(A, C) | C \in \Omega_\ell(B)\}$ (it is worth noting that under this interpretation, $d_\ell(A, B) = d_\ell(B, A)$).

2.5 A pictorial interpretation of a geodesic between two points on the Grassmannian

In this section, we present a more intuitive interpretation of the logarithmic map introduced in Section 2.4.2. The computational formula derivation in Section 2.4.2 is merely a sequence of algebraic computation. Here we will give a more intuitive picture and proof to show that the geodesic path between two points on the Grassmannian, is exactly a path between the principal vectors of corresponding linear subspaces.

2.5.1 Geodesic as a path between principal vectors

Let X and Y denote two $n \times k$ matrices which represent two k-dimensional linear subspaces $\mathbf{X}$ and $\mathbf{Y}$ in $\mathbb{R}^n$ respectively. The principal angles $\Theta = diag\{[\theta_1, \theta_2, \cdots, \theta_k]\}$ can be computed via SVD of $X^T Y$:

$$V \Sigma U^T = X^T Y \tag{2.10}$$

where $\Theta = \arccos \Sigma$. Therefore, Equation (2.10) can be written as

$$X^T Y = V \cos\Theta U^T \tag{2.11}$$

Now if we rotate X and Y by U and V, we can obtain canonical basis

$$(XV)^T (YU) = \cos\Theta. \tag{2.12}$$

Here we define $X_c = XV$ and $Y_c = YU$ as the *canonical basis* for $\mathbf{X}$ and $\mathbf{Y}$. It is important to note that the i-th column of X_c and Y_c is the i-th pair of principal vectors corresponding to the i-th principal angle θ_i.

Let x_i and y_i denote the i-th column of canonical basis X_c and Y_c. We want to find the curve $\phi_i(t) : [0,1] \mapsto \mathbb{R}^n$ such that $\phi_i(t)$ moves from x_i to y_i in unit time. Let $\phi_i(t) = a_t x_i + b_t y_i$ be a vector in $\mathbb{R}^n$ of unit length. The angle between $\phi_i(t)$ and x_i is $t\theta$ and the angle between $\phi_i(t)$ and y_i is $(1-t)\theta$. An illustration can be found in Figure 2.3. One can compute a_t and b_t by solving the following set of equations:

$$(a_t x_i + b_t y_i) \cdot x_i = cos(t\theta_i)$$
$$(a_t x_i + b_t y_i) \cdot y_i = cos((1-t)\theta_i)$$

It can be readily verified that $a_t = \cos(t\theta_i) - \dfrac{\cos\theta_i}{\sin\theta_i}\sin(t\theta_i)$ and $b_t = \dfrac{1}{\sin\theta_i}\sin(t\theta_i)$ and therefore

$$\phi_i(t) = x_i \cos(t\theta_i) + [y_i \frac{1}{\sin(\theta_i)} - x_i \frac{\cos(\theta_i)}{sin(\theta_i)}] \sin(t\theta_i)$$

Hence we can also find the corresponding matrix form

$$\Phi(t) = XV\cos(t\Theta) + [YU(\sin\Theta)^{-1} - XV(\tan\Theta)^{-1}]\sin(t\Theta) \tag{2.13}$$

where $V\Sigma U^T = X^T Y$ and $\Theta = \arccos\Sigma$.

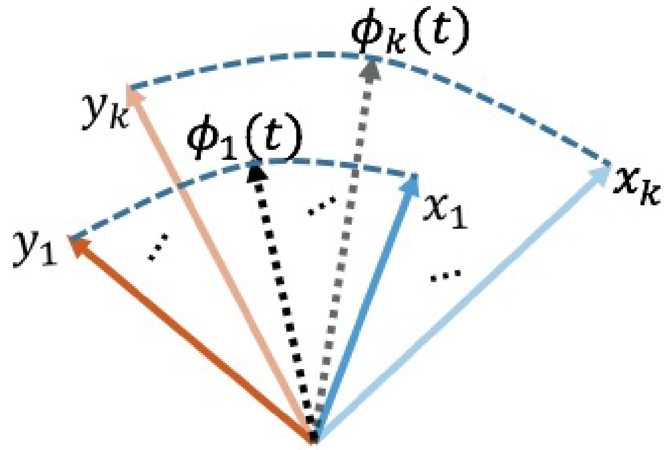


Figure 2.3: A pictorial understanding of a geodesic on the Grassmannian. x_i and y_i are the i-th pair principal vectors between 2 subspaces.

Claim 2.5.1. *The path between principal vectors defined in Equation* (2.13) *is exactly the geodesic path between k-dimensional subspaces* $\mathbf{X}$ *and* $\mathbf{Y}$.

The proof of this claim is given in Section 2.5.2.

2.5.2 Revisit logarithmic map

To prove Claim 2.5.1, let us revisit the derivation of the geodesic formula (2.17) given in Section 2.5.2.

Rewrite Equation (2.14) by multiplying D^T on both sides, we get

$$V \cos(\Theta)(DV)^T = X^T Y. \tag{2.14}$$

Due to properties of the $singular\ value\ decomposition$, the decomposition in Equation (2.14) is exactly the SVD defined in Equation (2.10) where $DV = U$. Equation (2.4) and (2.5) can be rewritten as:

$$V\cos(\Theta)U^T = X^T Y \tag{2.15}$$

$$M\sin(\Theta)U^T = (I - XX^T)Y \tag{2.16}$$

Then,

$$\begin{aligned} M\sin(\Theta)V^T(V\cos(\Theta)V^T)^{-1} &= M\tan(\Theta)V^T \\ &= (I - XX^T)Y(X^TY)^{-1} \end{aligned}$$

Therefore, to find the velocity matrix H, it suffices to compute the thin SVD, $(I-XX^T)Y(X^TY)^{-1} = M\Sigma V^T$, and $H = M\Theta V^T$ where $\Theta = \arctan(\Sigma)$. One subtlety in Equation (2.3) is that if V is multiplied from the right on both sides of the equation, we still have a representative of the same equivalence class as $\Phi(t)$, i.e. $\Phi(t)V$ is equivalent to $\Phi(t)$ on $Gr(k,n)$ for all t. With this in mind, we provide the formula which computes the shortest path between X and Y representing k-dimensional subspaces $\mathbf{X}$ and $\mathbf{Y}$,

$$G(t) = XV\cos\Theta t + M\sin\Theta t \tag{2.17}$$

The quantities V, Θ, and M can be found in the derivation above.

To prove Claim 2.5.1, we expand the right-hand side of Equation (2.16) and get

$$\begin{aligned} M\sin(\Theta)U^T &= Y - XX^TY \\ &= Y - XV\cos(\Theta)U^T \end{aligned}$$

Multiply both sides by $U(\sin\Theta)^{-1}$ to yield:

$$M = YU(\sin\Theta)^{-1} - XV(\tan\Theta)^{-1} \tag{2.18}$$

By substituting Equation (2.18) into Equation (2.17), we observe that Equation (2.13) and (2.17) are identical.

2.5.3 Summary

The proof given in this section leads to a pictorial understanding of the geodesic on the Grassmannian, i.e. the geodesic path between two k-dimensional linear subspaces is identical to the path between the principal vectors of the subspaces. As illustrated in Figure 2.3, moving one subspace to another, in other words, moving from one point to another on the Grassmannian, is simply moving the corresponding pairwise principal vectors.

Chapter 3

A geometric approach to missing data problem

3.1 Introduction

The major question we want to address in this chapter is that, given a data set with missing (or corrupted) entries, assuming we know the location of missing data, is it possible to recover the data set? For example, given a set of traveling waves with a certain percentage of data missing, can the missing entries be imputed from the rest of the data set? Another example would be, one may have a set of images of faces, where each picture has some corrupted pixels and one wants to impute the missing entries. In [20], Everson and Sirovich presented a popular iterative procedure to impute missing entries in a dataset, which is also referred to as gappy POD (proper orthogonal decomposition) in [52]. The idea of gappy POD is to minimize error in an iterative way in the sense that in each step, the objective function one is trying to minimize is changing. On the contrary, we propose a single energy function that the Everson and Sirovich approach is attempting to optimize. More importantly, one may now solve the minimization problem by various gradient-based descent algorithms. One important consequence of posing an energy function minimization problem is that we can include regularization terms if necessary, which could potentially help us avoid local minima and improve performance. Indeed, we show that the fixed points of the iterative procedure are local minima of this energy function, and also show that fixed points of the iterative procedure exist. We also give a proof that with proper embedding, our energy function along with a selected distance metric could potentially help us obtain an analytical solution to the minimization problem.

We provide background material and notation in Section 3.2 and describe the Everson and Sirovich's iterative procedure in Section 3.3. In Section 3.4 we recharacterize this iterative procedure as an energy minimization on the Grassmann manifold, and in Section 3.5 we solve the energy minimization problem via Grassmannian steepest gradient descent algorithm. We provide examples on real data in Section 3.9, including an example on time series data where Everson and Sirovich's iterative procedure gets stuck in a local minimum, whereas gradient descent converges to the global minimum. Finally, in Section 3.10, it is shown that with proper embedding, our optimization problem can be rewritten and solved analytically. This solution is closely related to the flag mean representation discussed in [18].

3.2 Notation

In this section, we will first introduce some notations we will use in this Chapter.

3.2.1 Data with missing entries

Let $X = \{x^{(i)} \mid i = 1, \ldots, P\} \subseteq \mathbb{R}^n$ be a collection of P points in $\mathbb{R}^n$. The data points may contain missing entries, which is recorded by a collection of masks $\{m^{(i)} \mid i = 1, \ldots, P\} \subseteq \{0, 1\}^n$, where

- if $m_j^{(i)} = 1$ then entry $x_j^{(i)}$ is present, and
- if $m_j^{(i)} = 0$ then the j-th entry of $x^{(i)}$ is missing.

For convenience, if the j-th entry of $x^{(i)}$ is missing, then we set $x_j^{(i)} = 0$.

Our goal is to impute the missing entries of the dataset X. These missing entries will be stored in vectors $t^{(i)}$, where if the j-th entry of $x^{(i)}$ is present then we set $t_j^{(i)} = 0$. The repair of a vector $x^{(i)}$ is equal to $x^{(i)} + t^{(i)}$, and hence the repair of dataset X is equal to $\{x^{(i)} + t^{(i)} \mid i = 1, \ldots, P\}$. Note that $x^{(i)}$ and $t^{(i)}$ are orthogonal.

For $y, z \in \mathbb{R}^n$, let $(y, z)_{m^{(i)}} = \sum_{\{j \mid m_j^{(i)}=1\}} y_j z_j$ denote the inner product with respect to mask $m^{(i)}$. For any vector y we have $(y, x^{(i)})_{m^{(i)}} = (y, x^{(i)})$ since we've set $x_j^{(i)} = 0$ if $m_j^{(i)} = 0$.

3.2.2 Distance between sets

Given two sets $S_1, S_2 \subseteq \mathbb{R}^n$, let $d(S_1, S_2) = \inf\{\|x_1 - x_2\| \mid x_1 \in S_1, x_2 \in S_2\}$ denote the infimum Euclidean distance between them.

3.2.3 The Grassmannian

Let $Gr(k,n)$ denote the set of all k-dimensional linear subspaces of $\mathbb{R}^n$. The Grassmannian $\mathrm{Gr}(k,n)$ is a manifold of dimension $k(n-k)$. We will often represent a k-dimensional linear subspace $\mathbf{V} \in \mathrm{Gr}(k,n)$ as a matrix $V \in \mathbb{R}^{n\times k}$ such that $V^TV = I$, even though this choice of matrix V is not unique. Note here we are viewing $Gr(k,n)$ as a quotient manifold of the Stiefel manifold $St(k,n)$.

To implement a gradient descent algorithm on $Gr(k,n)$, we recall two essential geometry elements.

- The gradient of a function on the Grassmanian. Let $f\colon Gr(k,n) \to \mathbb{R}$ be a real-valued function defined on the Grassmannian, and let f_V be the partial derivative of f with respect to $V \in \mathbb{R}^{n\times k}$. Then the gradient of f at $\mathbf{V}$ is

$$\nabla f = f_V - VV^T f_V.$$

- Retraction at a point $V \in \mathrm{Gr}(k,n)$. For a gradient descent algorithm defined on a manifold, it is important to be able to move on the manifold from an initial point V in a given direction H. Here we define the retraction $R_V(H)$ algorithmically,

$$R_V(H) = \mathbf{qr}(V+H,0),$$

  where $\mathbf{qr}(V+H,0)$ is the Q-component of the QR decomposition of $V+H$.

3.3 Everson and Sirovich's iterative procedure (gappy POD)

In [20], Everson and Sirovich propose an iterative algorithm for imputing the missing entries in a dataset $X \subseteq \mathbb{R}^n$. This method is based on the Karhunen–Loève or PCA procedure to model a dataset $X \subseteq \mathbb{R}^n$ using a k-dimensional linear subspace, where k is fixed. First, each missing entry in dataset X is imputed via an initial guess—for example, one may initially impute a missing entry $x_j^{(i)}$ to be the average of all the entries $x_j^{(i')}$ as i' varies over all data points in which the j-th entry is present. After these initial guesses have been made, one then applies the following iterative procedure:

- A k-dimensional linear model is computed to best fit the current dataset.
- The missing entries of each data point are then imputed to move that data point as close as possible to the k-dimensional linear model.

One iteratively computes a k-dimensional model on the new data points, and then moves each data point as close as possible to that model, until a satisfactory level of convergence has been reached.

3.4 Characterization as an optimization problem

Given a dataset $X \subseteq \mathbb{R}^n$ with missing entries, let $d(i) = n - \|m^{(i)}\|_1$ be the number of missing entries in the i-th data point $x^{(i)}$, and let $\{j \mid m_j^{(i)} = 0\} = \{j_1, j_2, \ldots, j_{d(i)}\}$ be the set of missing entries in this data point. Let e_k be the k-th standard basis vector in $\mathbb{R}^n$. Each data point $x^{(i)}$ corresponds to a coordinate-aligned $d(i)$-dimensional affine subspace

$$A_i = \left\{ x^{(i)} + t^{(i)} \;\middle|\; t^{(i)} \in \operatorname{span}(e_{j_1}, \ldots, e_{j_{d(i)}}) \right\},$$

which is the set of all possible repair vectors.

Given $1 \leq k \leq n$ fixed, our proposed optimization problem is to find a k-dimensional linear subspace $\mathbf{V}^*$ of $\mathbb{R}^n$ (i.e. $\mathbf{V}^* \in \mathrm{Gr}(k, n)$) solving the following optimization problem:

$$\mathbf{V}^* = \arg \min_{\mathbf{V} \in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, A_i). \tag{3.1}$$

That is, we want to find the linear subspace $\mathbf{V}^*$ which minimizes the sum of squared distances to each coordinate-aligned affine space A_i.

Define the objective function $f \colon \mathrm{Gr}(k, n) \to \mathbb{R}$ by

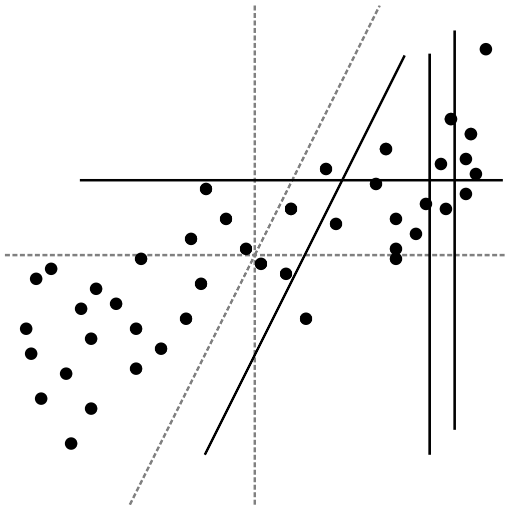


Figure 3.1: (Left) A data point with no missing entries is represented as a point, and a data point with one missing entry is represented as a line. For example, the two data points with missing y-values are represented as vertical lines. More generally, a data point $x^{(i)}$ with $k(i)$ missing entries is represented as a coordinate-aligned $k(i)$-dimensional affine subspace A_i. (Right) In blue we have drawn the optimal 2-dimensional linear subspace $\mathbf{V}^*$ minimizing the sum of squared distances to the data subspaces A_i. The repaired data points with imputed missing values are drawn in red.

$$f(\mathbf{V}) = \sum_{i=1}^{P} d^2(\mathbf{V}, A_i). \tag{3.2}$$

Theorem 3.4.1. *If $\mathbf{V}^*$ solves the optimization problem* (3.1)*, and if each data point $x^{(i)}$ is repaired to be a vector $x^{(i)} + t^{(i)} \in A_i$ satisfying $d(\mathbf{V}^*, x^{(i)} + t^{(i)}) = d(\mathbf{V}^*, A_i)$, then the repaired dataset $\{x^{(i)} + t^{(i)} \mid i = 1, \ldots, P\}$ is a fixed point of Everson and Sirovich's iterative procedure in [20].*

Proof. Suppose $\mathbf{V}^* \in \mathrm{Gr}(k, n)$ solves the optimization (3.1), and suppose each data point $x^{(i)}$ is repaired to be a vector $x^{(i)} + t^{(i)} \in A_i$ satisfying $d(\mathbf{V}^*, x^{(i)} + t^{(i)}) = d(\mathbf{V}^*, A_i)$. Note

$$\mathbf{V}^* = \arg \min_{\mathbf{V} \in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, A_i) = \arg \min_{\mathbf{V} \in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, x^{(i)} + t^{(i)}),$$

and hence when we compute a k-dimensional PCA hyperplane approximation for dataset $\{x^{(i)} + t^{(i)} \mid i = 1, \ldots, P\}$, this hyperplane can be chosen to be $\mathbf{V}^*$ (which minimizes the mean squared error). When we repair the dataset $\{x^{(i)} + t^{(i)} \mid i = 1, \ldots, P\}$ using hyperplane $\mathbf{V}^*$, by definition of

$t^{(i)}$ this repaired dataset can be chosen to be unchanged. It follows that $\{x^{(i)} + t^{(i)} \mid i = 1, \ldots, P\}$ is a fixed point of Everson and Sirovich's iterative procedure. $\square$

One can use Theorem 3.4.1 to prove that fixed points of Everson and Sirovich's iterative procedure exist.

Theorem 3.4.2. *Given a dataset $X \subseteq \mathbb{R}^n$ with missing entries, there exists a fixed point of Everson and Sirovich's iterative procedure in [20].*

We expect that finding a global solution to the optimization problem (3.1) is not easy in all cases. However, Everson and Sirovich's iterative procedure will converge to some local minimum.

Proposition 3.4.3. *Let $\mathbf{V}[1], \mathbf{V}[2], \mathbf{V}[3], \ldots$ be the sequence of PCA hyperplanes computed as one solves Everson and Sirovich's iterative procedure in [20], and let $X[1], X[2], X[3], \ldots$ be the sequence of imputed datasets. If $F[j] := \sum_{i=1}^{P} d^2(\mathbf{V}[j], X[j]^{(i)})$, where $X[j]^{(i)}$ is the i-th data point in the j-th dataset, then the sequence of real numbers $F[1], F[2], F[3], \ldots$ is monotonically non-increasing.*

Proof. The monotonically non-increasing property is easy to verify since updating the dataset can only decrease the sum of the squared distances, and since by the properties of PCA the same is true upon updating the linear model.[1] $\square$

We remark that though other methods of solving (3.1) need not have this same monotonicity property, for example, if one performs gradient descent with step sizes that are too large. However, it is possible that other methods in some cases may be better at avoiding local minima.

3.5 Solving the optimization

In this section, we will discuss how to solve the optimization problem (3.1).

[1]Though the energy is monotonically converging, the imputed data points and the k-dimensional model are likely to be converging to a fixed point, though we haven't ruled out the case that they are instead converging to a periodic orbit.

3.5.1 The analytical expression for distance

Let $\mathbf{V} \in \mathrm{Gr}(k, n)$ denote a k-dimensional linear subspace, which can be represented as an orthonormal matrix $V \in \mathbb{R}^{n\times k}$ such that $V^T V = I$. That is, the columns of V form a set of k orthonormal vectors in $\mathbb{R}^n$. Any $\mathbf{V} \in \mathrm{Gr}(k, n)$ has several different matrix representations V. Let E_i denote a matrix of size $n \times d(i)$ with columns $e_{j_1}, e_{j_2}, \ldots, e_{j_{d(i)}}$. Note that

- A_i is the set of all vectors of the form $x^{(i)} + t^{(i)}$, where $t^{(i)}$ is in the column span of E_i.
- $E_i E_i^T$ is a diagonal $n \times n$ matrix with entry (j, j) equal to one if $x^{(i)}$ is missing entry j, and zero otherwise.
- $V_{m(i)} = (I - E_i E_i^T)V$
- $P_i = V_{m(i)}^T V_{m(i)}$

The distance between the linear subspace $\mathbf{V}$ and the affine subspace A_i can be expressed as the following minimization :

$$\begin{aligned} d^2(\mathbf{V}, A_i) &= \min_{a\in\mathbb{R}^k, b\in\mathbb{R}^{d(i)}} \|Va - x^{(i)} - E_i b\|^2 \\ &= \min_{a\in\mathbb{R}^k, b\in\mathbb{R}^{d(i)}} (Va - x^{(i)} - E_i b)^T (Va - x^{(i)} - E_i b). \end{aligned} \tag{3.3}$$

One may solve this optimization by setting $\dfrac{\partial d^2(\mathbf{V}, A_i)}{\partial a}$ and $\dfrac{\partial d^2(\mathbf{V}, A_i)}{\partial b}$ equal to 0. Take the partial derivative of $d^2(\mathbf{V}, A_i)$ with respect to a and b and get

$$\begin{aligned} \frac{\partial d^2(\mathbf{V}, A_i)}{\partial a} &= 2a - 2V^T E_i b - 2V^T x^{(i)} \\ \frac{\partial d^2(\mathbf{V}, A_i)}{\partial b} &= -2E_i^T Va + 2b. \end{aligned}$$

Set both equations equal to zero and solve for a and b,

$$a = (I - V^T E_i E_i^T V)^{\dagger} V^T x_i \tag{3.4}$$

$$b = E_i^T V a \tag{3.5}$$

where $V_{m(i)} = (I - E_i E_i^T)V$. Note $I - E_i E_i^T$ is a projection matrix that projects V onto the orthogonal complement of E_i. The resulting $V_{m(i)}$ is erasing the rows of V associated with the missing entries of i^{th} data point.

Note that the matrix $I - V^T E_i E_i^T V$ is invertible whenever $\mathbf{V}$ does not contain the line $\langle e_j \rangle$ for any standard basis vector e_j corresponding to a missing coordinate of $x^{(i)}$, in which case our computation simplifies[2]. Let $P_i = I - V^T E_i E_i^T V$, we can compute the differential and get,

$$\begin{aligned}
\partial d^2(\mathbf{V}, A_i) &= -\mathrm{Tr}[x_i^T \partial V P_i^{-1} V^T x_i + x_i^T V \partial (P_i^{-1} V^T x_i)] \\
&= -\mathrm{Tr}[x_i^T \partial V P_i^{-1} V^T x_i + x_i^T V (\partial P_i^{-1}) V^T x_i + x_i^T V P_i^{-1} \partial V^T x_i] \\
&= -\mathrm{Tr}[x_i^T (\partial V) P_i^{-1} V^T x_i] + \mathrm{Tr}[x_i^T V P_i^{-1} (\partial P_i) P_i^{-1} V^T x_i] - \mathrm{Tr}[x_i^T V P_i^{-1} \partial V^T x_i].
\end{aligned}$$

Here $\partial P_i = \partial(I - V^T E_i E_i^T V) = -((\partial V^T) E_i E_i^T V + V^T E_i E_i^T (\partial V))$. Therefore, the middle term in the differential is

$$\begin{aligned}
&\mathrm{Tr}[x_i^T V P_i^{-1} (\partial P_i) P_i^{-1} V^T x_i] \\
=&\mathrm{Tr}[x_i^T V P_i^{-1} (-((\partial V^T) E_i E_i^T V + V^T E_i E_i^T (\partial V))) P_i^{-1} V^T x_i] \\
=&-\mathrm{Tr}[x_i^T V P_i^{-1} (\partial V^T) E_i E_i^T V P_i^{-1} V^T x_i] - \mathrm{Tr}[x_i^T V P_i^{-1} V^T E_i E_i^T (\partial V) P_i^{-1} V^T x_i].
\end{aligned}$$

Since the trace of a matrix (a product of matrices) is invariant under transpose and cyclic permutation, we rearrange the differential to get

[2]In the case where $I - V^T E_i E_i^T V$ is invertible, we can compute the corresponding differential to obtain

$$\begin{aligned}
\partial d^2(\mathbf{V}, A_i) &= \partial(x_i^T x_i - x_i^T V (I - V^T E_i E_i^T V)^{-1} V^T) x_i) \\
&= -\partial(x_i^T V (I - V^T E_i E_i^T V)^{-1} V^T) x_i) \\
&= -\partial \, \mathrm{Tr}(x_i^T V (I - V^T E_i E_i^T V)^{-1} V^T) x_i).
\end{aligned}$$

$$\partial d^2(\mathbf{V}, A_i) = -2\text{Tr}[(E_i E_i^T V P_i^{-1} V^T x_i x_i^T V P_i^{-1})\partial V^T] - 2\text{Tr}[(x_i x_i^T V P_i^{-1})\partial V^T].$$

The gradient of $d^2(\mathbf{V}, A_i)$ is

$$\nabla d^2(\mathbf{V}, A_i) = -2x_i x_i^T V P_i^{-1} - 2E_i E_i^T V P_i^{-1} V^T x_i x_i^T V P_i^{-1}. \tag{3.6}$$

Therefore, the Grassmannian gradient of the objective function (3.2) at V is:

$$\nabla f = -2(I - VV^T) \sum_{i=1}^{P} (x_i x_i^T V P_i^{-1} - 2E_i E_i^T V P_i^{-1} V^T x_i x_i^T V P_i^{-1}) \tag{3.7}$$

Algorithm 2: Grassmannian($\text{Gr}(k, n)$) steepest gradient descent algorithm

Input Data: Initial position: $\text{V}[0]$, Objective function: $\text{f}(\text{V})$, Gradient of f: ∇f, Step size:ϵ
Output Data: Optimal basis $\mathbf{V}^*$
Result: (Local) optimum basis for solving the optimization :

$$\mathbf{V}^* = \arg \min_{\mathbf{V} \in \text{Gr}(k,n)} \sum_{i=1}^{P} \|[I - V_{m(i)}(V_{m(i)}^T V_{m(i)})^{-1} V_{m(i)}^T] x_i\|^2$$

For: $j = 1, 2, \ldots$
Step 1: Move along negative gradient direction: $\text{V}[\text{j}-1] = \text{V}[\text{j}-1] - \epsilon\nabla\text{f}(\text{V}[\text{j}-1])$
Step 2: Retraction back to $\text{Gr}(k, n)$: $\text{V}[\text{j}] = \mathbf{qr}(\text{V}[\text{j}-1])$
Step 3: Iterate

3.6 Relationship to Riemannian matrix completion

We propose a Grassmannian optimization approach to solve the missing data problem, which we start with a geometric interpretation. We also notice that our minimization over the sum of the distances between $\mathbf{V}$ and affine subspaces A_i is closely related to matrix completion, especially Riemannian matrix completion [46] and GROUSE (Grassmannian Rank-One Update Subspace Estimation) [6]. Here we detail the relationship to [46]. The following are notations from [46]:

- P_Ω is the projection onto the entries where the data is present.

- X is the rank k matrix to find, with i-th column X_i.
- A is the data matrix.
- $P_\Omega(A)_i$ is the i-th column ($P_\Omega(A)_i$ corresponds to the i-th data point).

The following are notations regularly used in this dissertation:

- $\mathbf{V} \in \mathrm{Gr}(k,n)$ is the dimension k linear subspace to find.
- Let vectors $v_1, \ldots, v_k$ be such that $\mathrm{span}(v_1, \ldots, v_k) = \mathbf{V}$.
- A_i is the coordinate-aligned subspace corresponding to the i-th data point.

See also [17,26,32]. The paper [5] cites Everson & Sirovich in the context of low rank approximation methods.

Here is a derivation, which eventually leads to the conclusion that the optimal solution (assuming existence) to our optimization (3.1) is also the solution to the optimization problem posed in [46]. The whole proof can be divided into three claims.

Claim 3.6.1. *We have*

$$\min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, A_i) = \min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \min_{C\in\mathbb{R}^{k\times m}} \|P_\Omega(VC) - P_\Omega(A)\|_F^2.$$

Proof. This can be shown by dividing the Frobenius norm on the right-hand side into the sum of l_2 norm of columns. First, we introduce some new notations:

- $P_{\Omega_i}(V)$: mask the rows of V corresponding to the missing entries of i^{th} data point
- C_i: i^{th} column of $\mathbf{C}$

$$
\begin{aligned}
&\min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \min_{\mathbf{C}\in\mathbb{R}^{k\times m}} \|P_\Omega(VC) - P_\Omega(A)\|_F^2 \\
&= \min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \sum_{i=1}^{P} \min_{\mathbf{C}\in\mathbb{R}^{k\times m}} \|P_\Omega(VC)_i - P_\Omega(A)_i\|^2 \\
&= \min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \sum_{i=1}^{P} \min_{C_i\in\mathbb{R}^{k\times 1}} \|P_{\Omega_i}(VC_i) - P_\Omega(A)_i\|^2 \\
&= \min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \sum_{i=1}^{P} \min_{C_i\in\mathbb{R}^{k\times 1}} \|P_{\Omega_i}(V)C_i - P_\Omega(A)_i\|^2 \\
&= \min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, A_i)
\end{aligned}
$$

The last equation can be verified by substituting (3.4) and (3.5) to (3.3). □

Claim 3.6.2. *We have*

$$
\min_{X\in\mathcal{M}_k} \|P_\Omega(X) - P_\Omega(A)\|_F^2 = \min_{\mathbf{V}\in\mathrm{Gr}(k,n),\ C\in\mathbb{R}^{k\times m}} \|P_\Omega(VC) - P_\Omega(A)\|_F^2,
$$

where $\mathcal{M}_k := \{X \mid rank(X) = k\}$.

Proof. This is simply optimizing over a factorization of X instead of optimizing over X itself. □

Claim 3.6.3. *Let V^* and C^* be the optimal solution to* $\min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \min_{C\in\mathbb{R}^{k\times m}} \|P_\Omega(VC) - P_\Omega(A)\|_F^2$. *Then V^* and C^* is also an optimal solution to* $\min_{\mathbf{V}\in\mathrm{Gr}(k,n), C\in\mathbb{R}^{k\times m}} \|P_\Omega(VC) - P_\Omega(A)\|_F^2$.

Proof. Let

$$
f(V, C) = \|P_\Omega(VC) - P_\Omega(A)\|_F^2
$$

and let V^* and C^* be an optimal solution to

$$
\min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \min_{C\in\mathbb{R}^{k\times m}} f(V, C).
$$

For any $\mathbf{V} \in \mathrm{Gr}(k, n)$ and any $\mathbf{C} \in \mathbb{R}^{k\times m}$, we have

$$
f(V, C) \geq \min_{C\in\mathbb{R}^{k\times m}} f(V, C) \geq \min_{\mathbf{V}\in\mathrm{Gr}(k,n)} \min_{\mathbf{C}\in\mathbb{R}^{k\times m}} f(V, C).
$$

Since

$$f(V^*, C^*) = \min_{\mathbf{V}\in \mathrm{Gr}(k,n)} \min_{C\in\mathbb{R}^{k\times m}} f(V, C),$$

we have

$$f(V, C) \geq f(V^*, C^*).$$

By definition, V^*, C^* is also an optimal solution to

$$\min_{\mathbf{V}\in \mathrm{Gr}(k,n),\ C\in\mathbb{R}^{k\times m}} \|P_\Omega(VC) - P_\Omega(A)\|_F^2.$$

□

We can also translate between the optimal rank k matrix X in [46], and our optimal linear subspace $\mathbf{V} \in \mathrm{Gr}(k, n)$. Indeed, given X, we obtain $\mathbf{V}$ by taking its column span. Vice-versa, given $\mathbf{V}$, we find the i-th column of X to be the closest point on $\mathbf{V}$ to the coordinate-aligned affine subspace A_i (this point is $X_i = \sum_{j=1}^k c_{i,j} v_j$).

3.7 Connection to Gappy POD

One might also associate the computation of $d^2(\mathbf{V}, A_i)$ to the process of recovering marred faces in [20]. In [20], Everson and Sirovich solved the following in Section 2: given an empirical orthonormal basis $V = [v_1|v_2|\cdots|v_k] \in \mathbb{R}^{n\times k}$ such that $V^T V = I$, how to reassemble a signal $x_i \in \mathbb{R}^n$ which is supported only on Ω_i? Everson and Sirovich proposed to find the best fit using a least-square criterion, i.e. minimize the error

$$\begin{aligned} E_i &= \|P_{\Omega_i}(x_i - \sum_{j=1}^{k} c_{i_j} v_j)\|^2 \\ &= \|P_{\Omega_i}(x_i) - P_{\Omega_i}(VC_i)\|^2 \end{aligned} \tag{3.8}$$

It can be verified that the solution to this minimization is obtained by solving the following system of linear equations

$$M_i C_i = f_i$$

for $C_i = [c_{i_1}, c_{i_2}, \cdots, c_{i_k}]^T$, where $M_i = V_{m(i)}^T V_{m(i)}$ and $f_i = V_{m(i)}^T x_i$. Assuming M_i is invertible, we can substitute $C_i = M_i^{-1} f_i$ back into Equation (3.8),

$$\begin{aligned}
E &= \|x_i - P_{\Omega_i}(V M_i^{-1} f_i)\|^2 \\
&= \|x_i - V_{m(i)}(V_{m(i)}^T V_{m(i)})^{-1} V_{m(i)}^T x_i\|^2 \\
&= \|(I - V_{m(i)}(V_{m(i)}^T V_{m(i)})^{-1} V_{m(i)}^T) x_i\|^2
\end{aligned}$$

which is exactly $d^2(\mathbf{V}, A_i)$, i.e., the distance between subspace $\mathbf{V}$ and affine subspace A_i. A natural question to ask at this moment is: what is the relationship between Gappy POD in [20] and our proposed optimization? One may view Gappy POD as an alternating algorithm for solving the optimization

$$\min_{\mathbf{V} \in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} \min_{a_i, b_i} \|V a_i - (x_i + E_i b_i)\|^2. \tag{3.9}$$

Recall that our approach solves the inner minimization problem analytically by fixing V, therefore (3.9) becomes an objective function of V only. The gappy POD procedure solves the inner and outer minimization alternatively by fixing V and a_i, b_i by turns. An algorithmic description of Gappy POD procedure is presented in Algorithm 3.

3.8 Optimization with a total variation penalty

Here we introduce a column-wise total variation penalty term to our energy function. Given a matrix $U \in \mathrm{Gr}(k, n)$, let $u_{i,j}$ denote the i-th row and j-th column of U. We define the column-wise total variation $\|U\|_{TV}^2$ as follows,

$$\|U\|_{TV}^2 = \sum_{j=1}^{k} \sum_{i=1}^{n-1} (u_{i+1,j} - u_{i,j})^2.$$

Algorithm 3: Gappy POD algorithm in the view of solving (3.1) in an alternating way

Input Data: Set of data points with missing entries $\{x_i\} \in \mathbb{R}^n$, $i = 1, \ldots, P$.
Output Data: Optimal basis $\mathbf{V}^*$
Result: Optimal basis for solving the optimization :

$$\mathbf{V}^* = \arg \min_{\mathbf{V} \in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} \min_{a_i, b_i} \|V a_i - (x_i + E_i b_i)\|^2$$

Initialization: Set k so that $\mathbf{V} \in \mathrm{Gr}(k,n)$, $x_i^{(0)} = \tilde{x}_i$ where the missing entries in $\tilde{x}_i$ is repaired by the average values of the other data points whose corresponding entry is available. $V^{(0)}$ is the first k columns of SVD basis of the column span of $[x_1^{(0)}|x_2^{(0)}|\cdots|x_P^{(0)}]$
For: $j = 0, 1, 2, \ldots$
Step 1: Given $V[j]$, minimize $\|V[j]a_i - (x_i[j] + E_i b_i)\|^2$ over a_i and b_i for each $i = 1, 2, 3, \cdots, P$, as the previous algebra work shows,
$x_i[j+1] = x_i[j] + (E_i E_i^T) V M_i^{-1} f_i$ where $M_i = V_{m(i)}^T V_{m(i)}$ and $f_i = V_{m(i)}^T x_i$
Step 2: Given $\{x_i[j+1]\}_{i=1}^P$, minimize $\sum_{i=1}^P d^2(V, x_i[j+1])$ over $V \in \mathrm{Gr}(k,n)$.
$V[j+1]$ is the first k columns of the SVD basis of the span of $\{x_1[j+1], x_2[j+1], \cdots, x_P[j+1]\}$
Step 3: Iterate

Our goal is to include this total variation penalty to our energy function. In order for it to make sense, we need to show that $\|U\|_{TV}^2$ is invariant to the change of basis, i.e., $\|U\|_{TV}^2 = \|UM\|_{TV}^2$ for any $k \times k$ orthogonal matrix M. It is easy to check that $\|U\|_{TV}^2 = \|LU\|_F^2$, where

$$L = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & -1 & 0 & & \\ 0 & 1 & -1 & & \\ & & & \ddots & 0 \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix} \in R^{n \times n}.$$

For any $k \times k$ orthogonal matrix M, we have

$$\begin{aligned} \|UM\|_{TV}^2 = \|LUM\|_F^2 = \mathrm{Tr}[(LUM)^T(LUM)] = \mathrm{Tr}[M^T U^T L^T L U M] \\ = \mathrm{Tr}[M M^T U^T L^T L U] = \mathrm{Tr}[U^T L^T L U] = \|LU\|_F^2 = \|U\|_{TV}^2. \end{aligned}$$

Hence we can modify the energy function to be

$$\min_{\mathbf{V}\in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, x^{(i)} + t^{(i)}) + \gamma \|V\|_{TV}^2, \tag{3.10}$$

which remains an optimization problem on the Grassmannian since $\|V\|_{TV}^2$ is orthogonally invariant. Here $\gamma \geq 0$ serves as a smoothness coefficient.

3.9 Energy minimization on example data sets

In this section we apply both the *Karhunen–Loève* (KL) procedure and our descent algorithm with and without total variation term to a synthetic sinusoidal wave dataset and facial images dataset. We observe that the KL-procedure at times gets stuck in a local minimum which our descent algorithm, including the total variation term, is at times able to avoid.

3.9.1 Sinusoidal wave

Our first application concerns the recovery of a synthetic incomplete sinusoidal wave dataset. Let the complete dataset be translationally invariant:

$$f(x_m, t_\mu) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{k} \sin[k(x_m - t_\mu)].$$

Here $m = 1, 2, \cdots, M$ with M the dimension of the ambient space (size of the spatial grid), and $\mu = 1, \cdots, P$ with P be the number of points in the ensemble. Let $x_m = (m-1)2\pi/M$ and let $t_\mu = (\mu - 1)2\pi/P$. We set $M = P = 64$ and $N = 2$ so the complete dataset is a 64×64 matrix with rank 4. All three algorithms are demonstrated on this ensemble of 64 sinusoidal waves, uniformly masked so that 80% of the data are missing. In Figure 3.2 an example is picked to show the difference between three algorithms. From the bottom right plot, we observe that both KL procedure and steepest gradient descent get stuck at a local minimum while gradient descent with total variation can find the global minimum and almost recover the incomplete dataset. The three other plots in Figure 3.2 illustrate the basis vectors obtained from each of the three algorithms; the basis functions found by descent with total variation are much smoother and closer to the (expected) trigonometric waves. Similar scenarios can be observed more frequently with a higher percentage ($\geq$ 80%) of missing entries. One thing to note, which can also be observed in Figure 3.2, is that the KL procedure generally converges to a local/global minimum faster than any of the gradient descent algorithms.

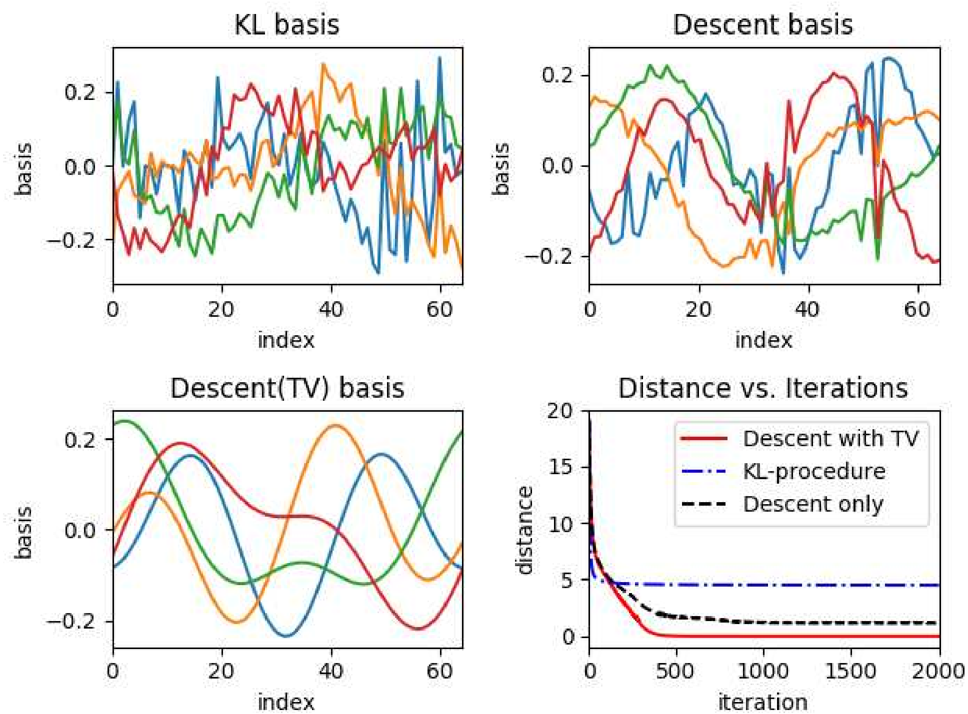


Figure 3.2: (Top Left) The basis obtained from 1000 iterations of KL-procedure in [20]. (Top Right) The basis obtained from solving (3.1) by Grassmannian gradient descent. (Bottom Left) The basis obtained from solving (3.10) by gradient descent. (Bottom Right) Function value defined in (3.1) versus the number of iterations across three methods.

Facial image recovery

To illustrate the utility of our proposed algorithms and compare them with the KL procedure, we apply them on a facial image dataset. An ensemble of 69 images of different people were used; in each image, the face has a still, neutral expression, with ambient lights on. Each image is processed via a level 4 Haar wavelet transform and the resolution is reduced to 90×68. The images are masked so that a fixed percentage of each face is obscured. An example of 40% missing pixels is shown in Figure 3.4. We concatenate the columns of each gray level intensity images so that each image is represented by a (vertical) vector of length 6120. Let $X = [x^{(1)}|x^{(2)}|\cdots|x^{(69)}]$ denote the whole facial image dataset, with each image vector $x^{(i)} \in \mathbb{R}^{6120}$. The masked data matrix is denoted by $X_m = [x_m^{(1)}|x_m^{(2)}|\cdots|x_m^{(69)}]$. One example of reconstructing an obscured image, using the KL procedure, steepest descent, and steepest descent with total variation is shown in Figure 3.4.

We introduce three measurements to quantify the performance of each algorithm.

1. Let $\tilde{x}^{(i)}$ denote the i^{th} reconstructed image. The reconstruction error $\|E\|$ is defined as $\|E\| = \sum_{i=1}^{69} \|\tilde{x}^{(i)} - x^{(i)}\|_2$.

2. Let $\mathbf{V}_{pca} \in \mathrm{Gr}(k, n)$ denote the k-dimensional PCA basis of the complete data matrix X, and let $\mathbf{V}_{rcon} \in \mathrm{Gr}(k, n)$ denote the reconstructed basis obtained from one of the algorithms. The Grassmannian geodesic distance is $d_{geo}(\mathbf{V}_{pca}, \mathbf{V}_{rcon}) = (\sum_{i=1}^{k} \theta_i^2)^{1/2}$ where $\theta_i \in [0, \pi/2]$ is the i^{th} principal angle between linear subspaces $\mathbf{V}_{pca}$ and $\mathbf{V}_{rcon}$. The geodesic distance is 0 if two linear subspaces are identical.

3. The commonality $Com(\mathbf{V}_{pca}, \mathbf{V}_{rcon})$ between linear subspaces $\mathbf{V}_{pca}$ and $\mathbf{V}_{rcon}$ is defined as $Com(\mathbf{V}_{pca}, \mathbf{V}_{rcon}) = \sum_{i=1}^{k} \cos^2(\theta_i)$, where θ_i is the i^{th} principal angle between $\mathbf{V}_{pca}$ and $\mathbf{V}_{rcon}$. Note that $Com(\mathbf{V}_{pca}, \mathbf{V}_{rcon}) = k$ if $\mathbf{V}_{pca}$ and $\mathbf{V}_{rcon}$ are identical, and that $Com(\mathbf{V}_{pca}, \mathbf{V}_{rcon}) = 0$ if $\mathbf{V}_{pca}$ and $\mathbf{V}_{rcon}$ are disjoint.

Figure 3.4 shows the measurements versus the rate at which the images were masked, where $\mathbf{V}_{pca}$, $\mathbf{V}_{rcon} \in \mathrm{Gr}(30, 6120)$. It is observed that in all three metrics, the pattern shows a similar trend, i.e., the descent algorithm to the optimization has a slight advantage over the gappy POD (KL procedure) algorithm while the optimization with total variation penalty shows better performance over the other two approaches. As more (higher fraction of) pixels are masked, the advantage of using a total variation penalty becomes more clear. The results are not surprising since facial image is largely smooth. The variation between pixels is usually subtle except for edges. Hence the algorithm is benefiting from promoting smoothness.

Figure 3.3: A demonstration of using various algorithm to fix face image data set.

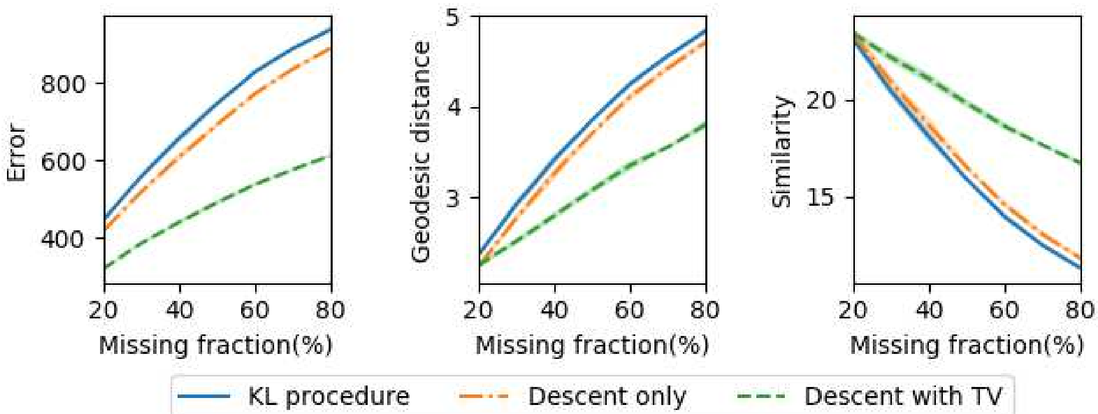


Figure 3.4: Performance comparison of different algorithms on the face image data set.

3.10 Affine Grassmannian and flag mean

In this section, we will introduce the affine Grassmannian and embed the coordinate aligned affine subspace A_i using the idea introduced in [35]. We will also show that when equipping with chordal distance, the optimal solution to our proposed minimization problem is closely related to the flag mean discussed in [18]. We will follow the notation in [35].

Definition 3.10.1. The Grassmannian of k-dimensional affine subspaces in $\mathbb{R}^n$, which is denoted by Graff(k,n), is the set of k-dimensional affine subspaces of $\mathbb{R}^n$.

Recall that we define coordinate-aligned affine subspace

$$A_i = \left\{ x^{(i)} + t^{(i)} \;\middle|\; t^{(i)} \in \operatorname{span}(e_{j_1}, \ldots, e_{j_{d(i)}}) \right\}.$$

Let $E_i = [e_{j_1}, \ldots, e_{j_{d(i)}}]$ and $\mathbf{E}_i$ denote the column span of E_i, one can also denote A_i as $\mathbf{E}_i + x^{(i)} \in$ Graff$(d(i), n)$, which can also be denoted by its *orthogonal affine coordinates* $[E_i, x^{(i)}] \in \mathbb{R}^{n\times(d(i)+1)}$ since E_i and $x^{(i)}$ are orthogonal ($E_i^T x^{(i)} = 0$).

The affine Grassmannian Graff(k, n) is an open submanifold of $Gr(k+1, n+1)$ along with an embedding which could be understood in the matrix setting. Let $[E_i, x^{(i)}]$ be the orthogonal affine coordinate of $\mathbf{E_i} + x^{(i)} \in \operatorname{Graff}(d(i), n)$, one can denote $\mathbf{E_i} + x^{(i)}$ by a $(n+1)$-by-$(d(i)+1)$ orthonormal matrix

$$Y_{E_i + x^{(i)}} = \begin{bmatrix} E_i & x^{(i)}/\sqrt{1 + \|x^{(0)}\|^2} \\ 0 & 1/\sqrt{1 + \|x^{(0)}\|^2} \end{bmatrix} \in St(d(i)+1, n+1).$$

A point on the affine Grassmannian, $\mathbf{A} + b \in \operatorname{Graff}(kn)$ can be represented as a set of $n+1$-by-$k+1$ orthonormal matrices in the equivalence class:

$$\left\{ \begin{bmatrix} A & b_0 \\ 0 & \gamma \end{bmatrix} \cdot M : \begin{bmatrix} A & b_0 \\ 0 & \gamma \end{bmatrix} \in St(k+1, n+1),\ M \in O(k+1) \right\},$$

where $b_0 = b/\sqrt{1 + \|b\|^2}$ and $\gamma = 1/\sqrt{1 + \|b\|^2}$. The affine Grassmannian thus can be viewed as the set of equivalence classes

$$\begin{aligned} \operatorname{Graff}(k, n) \simeq & \left\{ \begin{bmatrix} A & b_0 \\ 0 & \gamma \end{bmatrix} \cdot M : \begin{bmatrix} A & b_0 \\ 0 & \gamma \end{bmatrix} \in St(k+1, n+1),\ M \in O(k+1) \right\} \\ & \subseteq Gr(k+1, n+1). \end{aligned}$$

The advantage of embedding $\operatorname{Graff}(k, n)$ in $Gr(k+1, n+1)$ is that one can now utilize the well studied geometric framework of regular Grassmannian. Recall our optimization problem:

$$\mathbf{V}^* = \arg\min_{\mathbf{V}\in \mathrm{Gr}(k,n)} \sum_{i=1}^{P} d^2(\mathbf{V}, A_i)$$

where $\mathbf{V}$ is a k-dimensional linear subspace and $A_i = x^{(i)} + \mathbf{E_i}$ is a $d(i)$-dimensional affine subspace and the distance metric we are using here is simply Euclidean distance. With the embedding introduced above, one can map $\mathbf{V}$ to $Gr(k+1, n+1)$via

$$Y_V = \begin{bmatrix} V & 0 \\ 0 & 1 \end{bmatrix} \in St(k+1, n+1),$$

and map affine subspace $A_i = \mathbf{E}_i + x^{(i)}$ as

$$Y_{E_i + x^{(i)}} = \begin{bmatrix} E_i & x^{(i)}/\sqrt{1+\|x^{(i)}\|^2} \\ 0 & 1/\sqrt{1+\|x^{(i)}\|^2} \end{bmatrix} \in St(d(i)+1, n+1).$$

Also we denote their corresponding points (linear subspace) on the Grassmannian by $\mathbf{Y_V} \in Gr(k+1, n+1)$ and $\mathbf{Y_{E_i+x^{(i)}}} \in Gr(d(i)+1, n+1)$ respectively. We can rewrite our optimization problem as

$$\mathbf{Y_V}^* = \arg\min_{\mathbf{Y_V}\in \mathrm{Gr}(k+1,n+1)} \sum_{i=1}^{P} d_g(\mathbf{Y_V}, \mathbf{Y_{E_i+x^{(i)}}})^2, \tag{3.11}$$

where we choose d_g to be the chordal distance metric:

$$d_g(\mathbf{Y_V}, \mathbf{Y_{E_i+x^{(i)}}})^2 = \sum_{j=1}^{l} \sin^2(\theta_j^{(i)})$$

where $\{\theta_j\}_{j=1}^{l}$ are the principal angles between $\mathbf{Y_V}$ and $\mathbf{Y_{E_i+x^{(i)}}}$. Now we can write our optimization problem as:

$$
\begin{aligned}
\mathbf{Y_V}^* &= \arg \min_{\mathbf{Y_V} \in \mathrm{Gr}(k+1,n+1)} \sum_{i=1}^{P} d_g(\mathbf{Y_V}, \mathbf{Y_{E_i+x^{(i)}}})^2 && (3.12)\\
&= \arg \min_{\mathbf{Y_V} \in \mathrm{Gr}(k+1,n+1)} \sum_{i=1}^{P} \sum_{j=1}^{l(i)} \sin^2(\theta_j^{(i)}) && (3.13)\\
&= \arg \max_{\mathbf{Y_V} \in \mathrm{Gr}(k+1,n+1)} \sum_{i=1}^{P} \sum_{j=1}^{l(i)} \cos^2(\theta_j^{(i)}) && (3.14)\\
&= \arg \max_{Y_V \in \mathrm{St}(k+1,n+1)} \sum_{i=1}^{P} \mathrm{Tr}(Y_V^T Y_{E_i+x^{(i)}} Y_{E_i+x^{(i)}}^T Y_V) && (3.15)\\
&= \arg \max_{Y_V \in \mathrm{St}(k+1,n+1)} \sum_{i=1}^{P} \mathrm{Tr}(Y_{E_i+x^{(i)}} Y_{E_i+x^{(i)}}^T Y_V Y_V^T) && (3.16)
\end{aligned}
$$

The last equation above is due to the cyclic property of trace. Note that

$$
Y_{E_i+x^{(i)}} Y_{E_i+x^{(i)}}^T = \begin{bmatrix} E_i E_i^T + x^{(i)} x^{(i)T}/(1+\|x^{(i)}\|^2) & x^{(i)}/(1+\|x^{(i)}\|^2) \\ x^{(i)T}/(1+\|x^{(i)}\|^2) & 1/(1+\|x^{(i)}\|^2)) \end{bmatrix},
$$

$$
Y_V Y_V^T = \begin{bmatrix} VV^T & 0 \\ 0 & 1 \end{bmatrix}.
$$

and consequently,

$$
Y_{E_i+x^{(i)}} Y_{E_i+x^{(i)}}^T VV^T = \begin{bmatrix} [E_i E_i^T + x^{(i)} x^{(i)T}/(1+\|x^{(i)}\|^2)]VV^T & x^{(i)}/(1+\|x^{(i)}\|^2) \\ x^{(i)T} VV^T/(1+\|x^{(i)}\|^2) & 1/(1+\|x^{(i)}\|^2)) \end{bmatrix}.
$$

Since the bottom right element is a constant, to solve our optimization problem we only focus on the top left term. Our optimization problem now becomes

$$V^* = \arg \max_{V \in \mathrm{St}(k,n)} \sum_{i=1}^{P} \mathrm{Tr}([E_i E_i^T + x^{(i)} x^{(i)T} / (1 + \|x^{(i)}\|^2)] V V^T) \tag{3.17}$$

$$= \arg \max_{V \in \mathrm{St}(k,n)} \mathrm{Tr}([\sum_{i=1}^{P} (E_i E_i^T + x^{(i)} x^{(i)T} / (1 + \|x^{(i)}\|^2))] V V^T). \tag{3.18}$$

To simplify the notation, let $A = \sum_{i=1}^{P} (E_i E_i^T + x^{(i)} x^{(i)T} / (1 + \|x^{(i)}\|^2))$, and we have

$$V^* = \arg \max_{V \in \mathrm{St}(k,n)} \mathrm{Tr}(A V V^T)$$

$$= \arg \max_{V \in \mathrm{St}(k,n)} \mathrm{Tr}(V^T A V)$$

The constraints $V \in \mathrm{St}(k,n)$ can be interpreted as $V^T V = I$. Let Λ be a k-by-k symmetric matrix of Lagrange multipliers for the constraints. The goal is to maximize

$$\mathrm{Tr}[V^T A V + \Lambda (V^T V - I)]$$

By differentiation, it can be shown that

$$\frac{\partial}{\partial V} \mathrm{Tr}(V^T A V) = (A^T + A) V$$

$$\frac{\partial}{\partial V} \mathrm{Tr}(\Lambda V^T V) = 2 V \Lambda$$

Since A is symmetric, we can find that V must satisfy,

$$A V = V \Lambda, V^T V = I.$$

One can quickly verify that the eigenvalue decomposition (EVD) of A provides a solution to our optimization problem (3.17), i.e. let $A = U \Sigma U^T$ be the eigen-decomposition of A, also denote the "first" k columns of U corresponding to the k largest eigenvalues of A by U_k, then

$$V^* = U_k.$$

The embedding of affine subspaces onto the Grassmannian does give us a connection between our geometric approach for missing data and the flag mean idea. One can view the solution to the optimization problem (3.11) as the flag mean of all embedded affine subspaces associated with data points(with missing entries). Practical applications utilizing this idea will be something to be discovered for future work.

3.11 Conclusion

Given a dataset with missing (or corrupted) entries, the paper [20] gives an iterative procedure for imputing the missing entries. We recast this iterative procedure as an optimization. One can now solve this optimization via a wide variety of techniques, including gradient descent. It is now also possible to add regularization terms (such as adding a preference for smooth solutions) when performing gradient descent. We provide an example of time-series data where the iterative procedure of [20] gets stuck in a suboptimal local minimum (though this is rare), whereas the gradient descent algorithm we propose converges to the global minimum.

Chapter 4

Self-organizing Mappings on Grassmannian

In this section we will discuss a powerful nonlinear dimensionality-reducing tool known as the Self-Organizing Mappings, or SOMs and more importantly, we will extend SOMs to the Grassmann manifold so that we can utilize some of the attractive features of set-to-set pattern.

4.1 Introduction

The Self-Organizing Mappings was first introduced by Kohonen in [28]. This method has proven to be a valuable tool for visualization in low-dimensional space, see [29, 31] for more details. To be specific, in [30] Kohonen shows some extensive applications of SOM including bioinformatics which we will also explore in Section 4.3 under the context of biological pathway analysis. The key idea of this approach is that the data points which are close in higher dimensional space should be represented as neighbors in low dimensional space. The pseudocode of standard SOM is given in Algorithm 4.

Algorithm 4: Self-organizing Mappings

Input Data: Set of data points $\{x^{(\mu)}\} \in \mathbb{R}^n$, $\mu = 1, \dots, P$.
Output Data: Updated centers $\{c_i\}$, $i \in \mathcal{I}$
Initialization: Initialize a set of center vectors $\{c_i\}$, $c_i \in \mathbb{R}^n$,$i \in \mathcal{I}$
For: $j = 0, 1, 2, \dots$
Step 1: Present a randomly selected data point $x^{(\mu)}$ to the network.
Step 2: Determine the winning center c_{i^*}
Step 3: Update all the centers using

$$\delta c_i = \epsilon h(i - i^*)(x^{(\mu)} - c_{i^*})$$

Step 4: Iterate

The critical ingredients in Self-organizing Mappings is the multiplicative term $h(x)$ in the update rule. This function serves as the ordering of the center indices based on the nearness of the winning centers. Also $\epsilon h(x)$ is referred to as the **localization term**. In this section we select Gaussian function as our $h(x)$, i.e.,

$$h(x) = \exp(-x^2/r^2).$$

This function is essentially saying that all the centers are updated at a rate proportional to the nearness of their index to the winning index. If $d(i, i^*)$ is small then h is large, while the centers whose index is far away from $i^*(d(i, i^*)$ is big) get updated very slightly. As a result, this produces a $self$-$organizing$ map, since the centers with neighboring indices are trained to capture neighboring inputs and map them as neighbors. That being said, we say a map is self-organizing if

- $x^{(\mu)}$ has winning center $\mathbf{c}_{i(\mu)}$, $i(\mu) \in \mathcal{I}$
- $x^{(\nu)}$ has winning center $\mathbf{c}_{i(\nu)}$, $i(\nu) \in \mathcal{I}$
- the distance $d(x^{(\mu)}, x^{(\nu)})$ is small,

then the distance between the winning indices $d(i(\mu), i(\nu))$ is also small. The Self-organizing mappings described above may also be considered as a tool of dimensional reduction. The reduction takes input data points in the domain and maps them to the nearest center which is associated with a unique spatial index living in the center index set $\mathcal{I}$, i.e.,

$$SOM : x^{(\mu)} \in \mathbb{R}^n \longmapsto c_{i^*} \in \mathbb{R}^n \longmapsto i^* \in \mathcal{I}$$

For Euclidean SOM, the pattern $x^{(\mu)}$ and c_i are both living in $\mathbb{R}^n$. The index set $\mathcal{I}$ for Self-organizing mappings may also be attached with a topological structure induced by the distance metric $d(.,.)$ defined on $\mathcal{I}$. For example, the indices can be associated with points on a line, a circle or a plane. In this thesis, we select a rectangular lattice set of points of size $m \times n$ as our $\mathcal{I}$.

4.2 Grassmannian SOM

In this section we will develop the extension of the SOM algorithm on Euclidean space to the Grassmann manifold, from which data points parametrize all k-dimensional subspaces of n-dimensional vector space. As can be seen from Algorithm 4, to implement the Self-organizing mappings on some ambient space other than Euclidean space(e.g. the Grassmann/Flag manifold), all we need is the following two ingredients:

1. A measure of distance between two data points: $d(x^{(i)}, x^{(j)})$
2. A path to move one data point towards another.

Both of these two ingredients are already discussed in Section 2.4. Given two linear subspaces(two points in $Gr(k, n)$), any distance metric is a function of the principal angles. A path between two points from $Gr(k, n)$ is given as the logarithmic map formula in Equation (2.17). Now we are ready to extend SOM algorithm to the Grassmann manifold.

Since the algorithm iteratively updates the initial centers we use a superscript to denote the value of c_i at the m-th iteration. Recall that the update equation for Euclidean SOM is given by:

$$c_i^{m+1} = c_i^m + \epsilon h(d(i, i^*))(x - c_i^m)$$

where i^* is the spatial index of winning center associated to pattern x, i.e.

$$i^* = \arg\min_i \|x - c_i^m\|_2.$$

The distance between x and c_i^m is the standard Euclidean distance since x,$c_i^m \in \mathbb{R}^n$. Also as mentioned above the localization term $h(x) = \exp(-s^2/\sigma^2)$ and the distance metric d which induces the topology on $\mathcal{I}$ is defined as the Euclidean norm $d(i, j) = \|i - j\|_2$.

On the Grassmann manifold data points are no longer vectors in $\mathbb{R}^n$ but rather k-dimensional subspaces of $\mathbb{R}^n$. We select an initial set of centers $\{\mathbf{C}_i\}$ where $i \in \mathcal{I}$ is the spatial index and $\mathbf{C}_i \in Gr(k, n)$. For a given subspace $\mathbf{X} \in Gr(k, n)$ we find the closest center from the set of centers $\{\mathbf{C}_i\}$ by

$$i^* = \arg\min_i d_g(X, C_i)$$

where the distance metric d_g is given by Equation (2.8). To move the centers towards subspace $\mathbf{X}$ by certain amount, we compute the geodesic between each center $\mathbf{C_i}$ and data point $\mathbf{X}$, as described in Section 2.4, by computing the thin SVD,

$$U\Sigma V^T = (I - C_i^T)C_i(X^T C_i)^{-1}$$

The localization term now becomes

$$t = \epsilon_n h_n(d(a_i, a_{i^*})).$$

The nearness function is taken as

$$h_n(s) = \exp(-s^2/\sigma_n^2)$$

where $\sigma_n = \sigma_0(1 - n/T)$ and $\epsilon_n = \epsilon_0(1 - n/T)$. Therefore the centers are updated along the geodesic between $\mathbf{X}$ and $\mathbf{X}$ $C_i(t)$, by moving from $C_i(0)$ to $C_i(t)$, where localization term t is based on the local neighborhoods of spatial indices h_n and learning step ϵ_n.

4.3 Numerical Results

In this section we apply Grassmannian SOM on both synthetic data and biological gene expression datasets.

4.3.1 Synthetic data

To start we will present is an illustrative example to check if our algorithm can capture the topology of straight line in high-dimensional Grassmannian on a 2-D square lattice. In this ap-

Algorithm 5: Generate uniformly distributed random point on $\mathrm{Gr}(k,n)$(MATLAB code)

Input Data: $k, n \in \mathbb{N}$, where $k \leq n$
Output Data: $Q \in \mathbb{R}^{n \times k}$ s.t. $Q^T Q = I$
Result:

```
Function uniformRandMat(k,n):
    M = rand(n, n);
    tempM = M(:,1:k);
    Q = qr(tempM,0);
    return Q
```

plication, we need *uniform* samples from $O(n)$, St(k,n), and Gr(k,n), which could be achieved by constructing an $n \times n$ matrix of independent and identically-distributed normal$[0,1]$ random variables, and QR factoring it for **Q**. The MATLAB code which generates uniform samples on $Gr(k,n)$ is presented in Algorithm 5. We generate $Q_1, \cdots, Q_4 \in \mathbb{R}^{10\times 2}$ using algorithm 5 with $k = 2$, $n = 10$. These matrices serve as representatives of linear subspaces in $\mathrm{Gr}(2,10)$.

Parametrize a "straight line" on $\mathrm{Gr}(k,n)$

Our first application is illustrative in the sense that we are going to use 1-D integer index set to capture the topology of a "straight line" on $\mathrm{Gr}(2,10)$, i.e. a geodesic path between Q_1 and Q_2. We generate the data by sampling ten point uniformly along the geodesic path:

$$\Phi(t) = Q_1 V \cos(\Theta t) + U \sin(\Theta t)$$

where $U\Sigma V^T = (I - Q_1 Q_1^T) Q_2 (Q_1^T Q_2)^{-1}$ and $\Theta = \arctan \Sigma$. This idea of uniformly sample along geodesic is utilized in geodesic based Domain Adaptation under the field of computer vision, where the information between two domains are conveyed via geodesic path, see [21, 22] for more details. The Grassmannian SOM here serves to sort the points on the geodesic connecting Q_1 and Q_2. Note that in this example, the order of data points is provided in the sampling process:

$$P_i = \Phi(t_i)\,, t_i = (i-1)/10,\ i = 1, \cdots, 10.$$

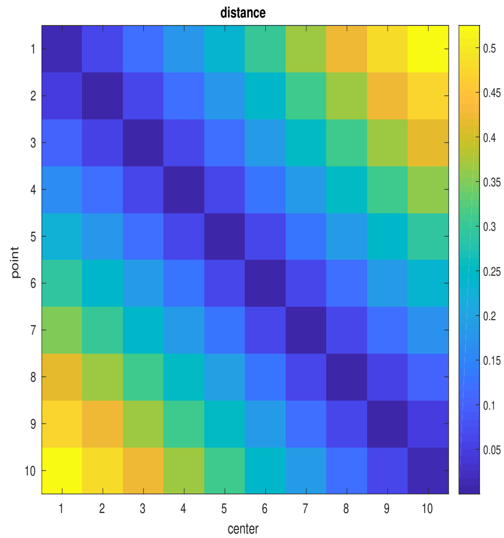


Figure 4.1: This matrix has the distances between the point i and center j after convergence. Note that point i is closest to center j when $i = j$, reflecting the ordering mechanism of the Grassmannian SOM.

Ten centers on $\mathrm{Gr}(2, 10)$ are randomly initialized using Algorithm 5. We use the integer index set $\{1, 2, \cdots, 10\}$ to capture the topology. The geodesic distance between points and the final order is shown in Figure 4.1.

4.3.2 Indian Pines

To illustrate the utility of the proposed method for visualizing real data, we apply it to the well-known Indian Pines hyperspectral image [34]. We have considered this data set before in the context of the band selection problem [13] and the persistent homology for signal detection on Grassmannians [15]. A related visualization application invokes the technique of multidimensional scaling and sparse support vector machines [14]. The classes are shown in Figure 5.

In this application we selected the 12 classes that were large enough to give 20 subspaces of dimension ten. Since this application is merely intended to illustrate the model we made no attempt to optimize our parameters. However, our previous work suggests these dimensions are reasonable [14]. Thus we are visualizing 240 labeled points in 220 dimensions by first constructing sets of 10-dimensional subspaces in 220-dimensions using the SVD.

We initialized the centers for Grassmannian-SOM by selecting 900 ten dimensional subspaces at random, corresponding to a 30×30 integer lattice. This was done by computing the singular value decomposition of matrices of size 220 by 10 from the uniform distribution. In Figure 4.2, we see the results of the Grassmannian-SOM algorithm where points in the same class have been organized to have similarly valued indices.

In Figures 4.3 and 4.4, we see the results of the Grassmannian-SOM when the points reside on $Gr(2, 220)$ and $Gr(1, 220)$, respectively. This data set is well-known as a challenging classification problem. For example, there are classes which are inherently very similar such as corn (green), corn-notill (red) and corn-mintill (blue). We see that these three classes are well-separated for SOM on $Gr(10, 220)$ while there is overlap using $Gr(1, 220)$ and $Gr(2, 220)$. In particular, the corn-mintill (blue) is much less localized on the lower-dimensional Grassmannians. We observe excellent clustering in the majority of classes with the possible exception of green pasture (x) which shows distinct spread suggesting it has significant spectral overlap with the other classes. These results vary the dimension of the Grassmannian and are higher resolution than those presented in the preliminary work [27].

4.3.3 Gene Expression Data

Here we examine the application of Grassmannian SOM to two gene expression data sets. The first is related to the immune response in mice to the Ebola virus while the second explores the human immune response to respiratory infection.

Ebola Mice Data

In this example we examine the application of Grassmannian SOM to a gene expression data set collected from mice responding to infection from the Ebola virus [41]. Each raw data point consists of a set of over 12,000 genes. As a preprocessing step we identify the subset of all discriminatory genes that classify infected versus controls; see [39,47] for details. Using these genes as the starting point we identified some 1300 biological pathways potentially of interest in the immune response to infection. Subsequently we applied machine learning techniques to select top pathways for

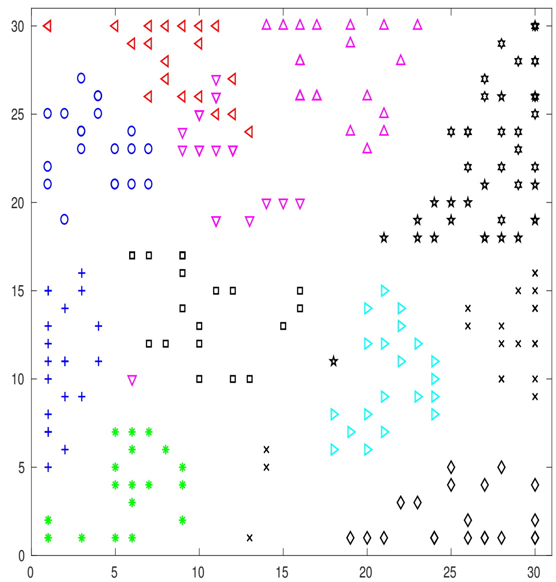


Figure 4.2: This figure shows the final configuration of the points as mapped to the 2D index set from $Gr(10, 220)$.

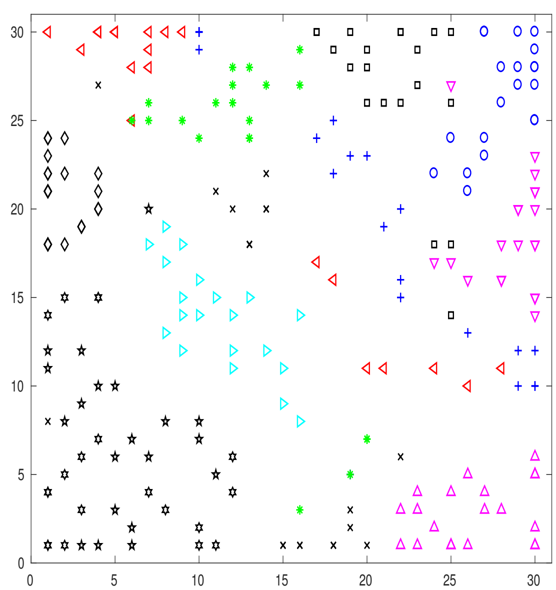


Figure 4.3: The converged Grassmannian SOM applied to the Indian Pines classes on $Gr(2, 220)$.

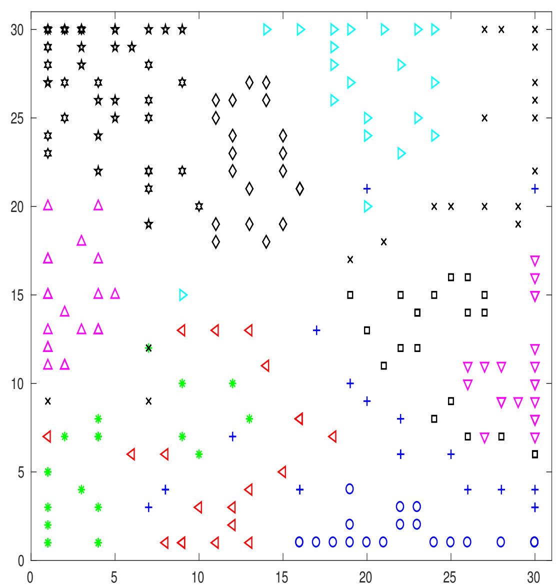


Figure 4.4: The converged Grassmannian SOM applied to the Indian Pines classes on $Gr(1, 220)$.

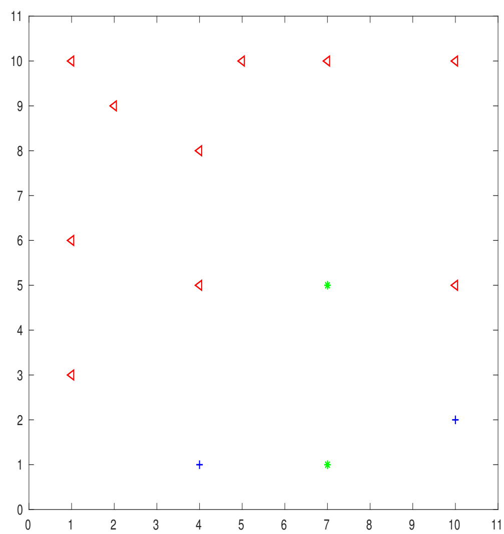


Figure 4.5: Mapping of gene expression data on T cell receptor signaling pathway.

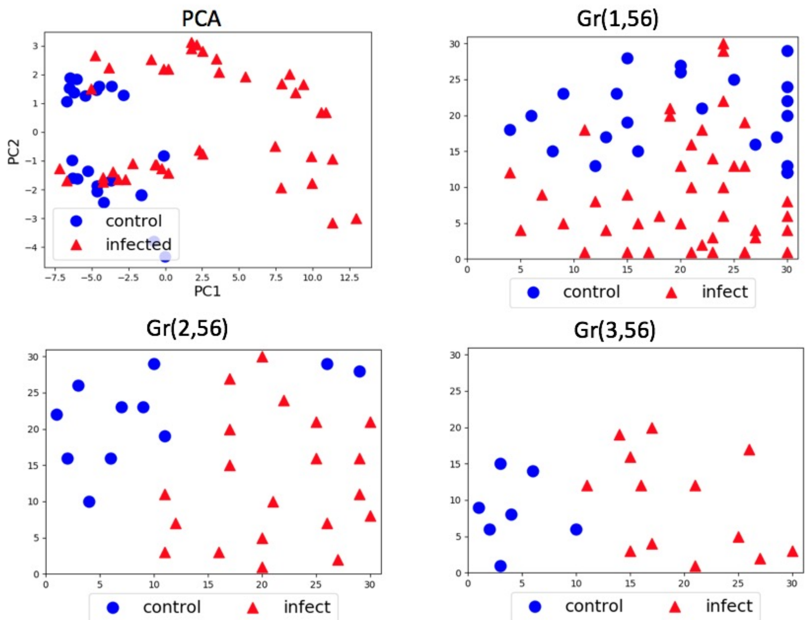


Figure 4.6: These four plots are 2D visualization of Uninfected Control and Infected subjects from hour 30 to 48. Top left: PCA visualization. Top right: Grassmannian-SOM on $Gr(1, 56)$. Bottom left: Grassmannian-SOM on $Gr(2, 56)$. Bottom right: Grassmannian-SOM on $Gr(3, 56)$.

further study. We have selected one of these pathways, i.e., the T cell receptor signaling pathway consisting of 48 genes, as an example to test the Grassmannian SOM algorithm on the Ebola Virus. We pick 3 points at random from each class to construct a single point on the Grassmannian. Hence each point on the Grassmannian lives on $Gr(3, 48)$, i.e., it consists of 48 genes and three biological samples. The result of training the Grassmannian SOM algorithm are shown in Figure 4.5 where the high dimensional observations are mapped to the two-dimensional index set in the usual manner. The red points represent control samples of healthy non-human primates. The green * points are the gene expression values at day one, while the blue + samples reflect expression at day 2 after exposure to infection by the Ebola virus. Although we do observe some of the desired clustering with this example, additional data appears to be required to provide a more complete picture. Hence, we present the following example on H3N2 influenza data set.

H3N2 Influenza Data

The H3N2 gene expression data sets was downloaded from GEO $GSE73072$ which consists of 7 studies. Two H3N2 challenge studies, i.e. $Dee2$ and $Dee5$ are selected for this experiment. Please see [37] for more details. We used the Reactome interferon alpha beta signaling pathway which contains 56 genes to form our data matrix, hence each data point resides on $Gr(k, 56)$. The solid blue circles represent uninfected control data (before inoculation) and red triangles represent infected samples from hour 30 to 48 after inoculation. For each k, we attached 900 randomly generated k dimensional subspaces to a 30×30 integer lattice, which is done in the same way as is described in section 4.3.2. In Figure 4.6, we see the results of Grassmannian-SOM when data points live on $Gr(1, 56)$(top right), $Gr(2, 56)$(bottom left) and $Gr(3, 56)$(right). We observe that two classes are well separated for SOM on $Gr(3, 56)$ while we start seeing overlaps on $Gr(2, 56)$ and even more overlaps on $Gr(1, 56)$. As a comparison, we also included the 2D visualization via PCA(top left) of this dataset, from which we can also find overlaps between two classes when data is projected onto the first two principal components. This example shows strong clustering performance when Grssmannian-SOM is applied to biological gene expression pathway data.

4.4 Conclusion

We have presented an extension of the self-organizing mapping algorithm to the geometric setting of the Grassmann manifold. The approach moves centers towards data points presented to the network by moving proportionally along the geodesic, or shortest path between two elements of $Gr(k, n)$. We illustrate the method by showing that the algorithm organizes the hyperspectral image data in the index space and separates ten dimensional subspaces of 220 dimensional space. While lower dimensional Grassmannians also capture significant structure, the 10-D subspaces captured the most variability consistent with observations made using other algorithms. We also observe that three dimensional subspaces resolve the H3N2 data into separable control and infected classes while these are clearly non-separable using either a standard PCA projection or Grassmannian SOM with one-dimensional subspaces. Hence the data subspace perspective is essential to adequately process the data using SOM.

Chapter 5

Extension to the Flag manifold

5.1 Overview of the flag manifold

The algorithms introduced in previous chapters are based on the Grassmann manifold. However some applications would require us to focus on a more generalized setting beyond a set of k-dimensional linear subspaces in $\mathbb{R}^n$. An interesting structure which generalizes the Grassmann manifold and captures additional geometry in data is known as the flag manifold. Intuitively, a flag is a sequence of nested subspaces. For example, a signal can be approximated with increasing resolution via a sequence of nested wavelet scaling space. This sequence of nested scaling subspaces is a point on the corresponding flag manifold. In addition, an ordered basis $v_1, v_2, \cdots, v_k$ of a data set obtained from principal component analysis also induces a sequence of nested subspaces $\mathbf{V_1} \subsetneq \mathbf{V_2} \subsetneq \cdots \subsetneq \mathbf{V_k}$ where $\mathbf{V_j} = span(v_1, \cdots, v_j)$. We first look at the definition of the flag manifold.

Definition 5.1.1. Let $\mathbf{p} = \{n_1, n_2, \cdots, n_d\}$ be a sequence of positive integers such that $\sum_{i=1}^{d} n_i = n$. A **flag**, $\mathbf{F}$, in $\mathbb{R}^n$ is a nested sequence of subspaces

$$\mathbf{V_1} \subsetneq \mathbf{V_2} \subsetneq \cdots \subsetneq \mathbf{V_d} \subsetneq \mathbb{R}^n$$

such that $dim(\mathbf{V_j}) = \sum_{i=1}^{j} n_i$. We denote the set of all such flags by $Fl(n_1, n_2, \cdots, n_d)$ and call it the flag manifold of type $\mathbf{p}$.

As a demonstrative example, a special case $Fl(1, 1, \cdots, 1)$, is the set of all flags of type $\{1, 1, \cdots, 1\}$, which is also referred to as a full flag. Figure 5.1 illustrates the first three nested subspaces of $Fl(1, 1, \cdots, 1)$, i.e. a line lives in a plane which lives in a 3-dimensional subspace. Also a flag of type $\{k, n-k\}$ is exactly a k-dimensional subspace in $\mathbb{R}^n$ or a point on the Grassmann manifold $Gr(k, n)$. In short, $Fl(k, n-k) = Gr(k, n)$. Here we introduce three cases where the nested(ordered) structure of flag naturally rises in data analysis.

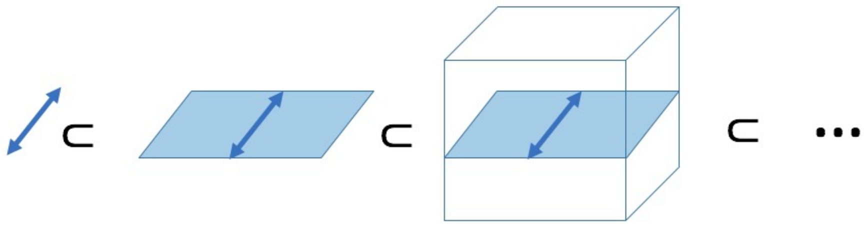


Figure 5.1: Illustration of a flag $Fl(1, 1, \cdots, 1)$, i.e. a 1-dimensional line living in a 2-dimensional plane living in a 3-dimensional space$\cdots$

1. Multi-resolution analysis: Wavelet analysis or Multi-resolution analysis is associated with a sequence of nested vector spaces which approximate data with increasing resolution. Each scaling subspaces $\mathbf{V_i}$ is a dilation of its neighboring subspace $\mathbf{V_{i+1}}$, i.e., if a signal $f(x) \in \mathbf{V_i}$ then a reduced resolution version $f(x/2) \in \mathbf{V_{j+1}}$. Here the scaling spaces are nested, i.e.,

$$\cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots .$$

In the finite dimensional setting, a nested scaling space can be understood as a flag on the corresponding flag manifold. For example, let $x = [1, 2, 3, 4]^T$ be a signal and we use Haar wavelet to approximate x. The Haar wavelet matrix H demonstrated in Figure 5.2 can be viewed as a point on $Fl(1, 1, 2)$ in the following way. The first scaling space $\mathbf{V_0}$ is the span of first column in H. The approximation of x using $\mathbf{V_0}$ is simply the average of four entries in x, i.e., a constant vector $\tilde{x}_0 = [2.5, 2.5, 2.5, 2.5]^T$, which can be understood as approximating x in its lowest resolution. Now if we use the span of the first two column of H, the second scaling space $\mathbf{V_{-1}}$ to approximate x(project x onto $\mathbf{V_{-1}}$), we end up with $\tilde{x}_{-1} = [1.5, 1.5, 3.5, 3.5]$. Now we are approximating x in a higher resolution. Finally, if we approximate x via the column span of H, which is the whole $\mathbf{V_{-2}} = \mathbb{R}^4$, we get x back. This can be viewed as approximating x under the finest resolution.

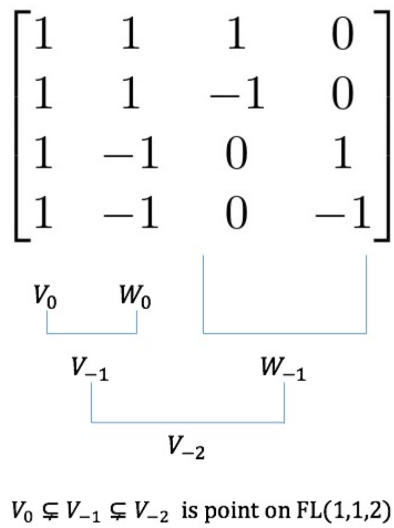


Figure 5.2: Illustration of D2 matrix and the associated nested scaling spaces

2. SVD basis of a real data matrix: Let $X \in \mathbb{R}^{n \times k}$ be an n-by-k real data matrix which consists of k-samples with n being the ambient dimension. And let $U\Sigma V^T = X$ be the compact SVD of X. U is an $n \times d$ orthonormal matrix consisting of d left singular vectors of $X$and more importantly, U is an ordered basis of the column span of X. The order of columns of U is determined by the magnitude of singular values of X. Putting this under the context of principal component analysis, the columns of U are associated to the ordered principal components and the order is determined by the amount of variance each principal vector captures from X. Let $U = [u_1|u_2|\cdots|u_d]$, then the nested subspaces $span(u_1) \subsetneq span(u_1, u_2) \subsetneq \cdots \subsetneq span(u_1, u_2, \cdots, u_d)$ is a flag of type $\mathbf{p} = \{1, \cdots, 1, n-d; n\}$ in $\mathbb{R}^n$. Later we will introduce the distance metric between two flags, from there one may consider the distance between two datasets by computing the distance between their SVD basis which takes the order of basis into account.

3. Aligned canonical basis corresponding to principal vectors: The geodesic between two k-dimensional subspaces can be understood as a path between corresponding principal vectors, which provides an order on both of two bases. To be more specific, Let $X, Y \in \mathbb{R}^{n\times k}$ be two orthonormal real matrices, the geodesic distance between the corresponding column span $\mathbf{X}, \mathbf{Y}$ can be computed by

$$d(\mathbf{X},\mathbf{Y}) = \sqrt{\sum_{i=1}^{k}\theta_i^2}$$

, where θ_i's are the singular values of X^TY:

$$X^TY = U\Sigma V^T, \Sigma = diag(\theta_1, \theta_2, \cdots, \theta_k).$$

Let $XU = [x_1|x_2|\cdots|x_d]$ and $YV = [y_1|y_2|\cdots|y_d]$, as already discussed in Section 2.5, x_i, y_i are i^{th} pair of principal vectors corresponding to subspaces $\mathbf{X}$ and $\mathbf{Y}$. Also XU and YV are known as the canonical basis of $\mathbf{X}$ and $\mathbf{Y}$. This order also induces a flag $span(x_1) \subsetneq span(x_1, x_2) \subsetneq \cdots \subsetneq span(x_1, x_2, \cdots, x_d)(span(y_1) \subsetneq span(y_1, y_2) \subsetneq \cdots \subsetneq span(y_1, y_2, \cdots, y_d))$ of type $\mathbf{p} = \{1, 1, \cdots, 1, n-k\}$.

5.2 The flag manifold as a matrix manifold

To implement any numerical algorithm on the flag manifold requires a matrix representation for points on the flag manifold. In this section, we will follow the idea in [19] and Section 2.4, we utilize the quotient manifold structure to represent flags and describe the tangent space.

5.2.1 Quotient manifold of $O(n)$

In Section 2.4, we follow the idea in [19] and describe the Grassmann manifold $Gr(k, n)$ as a quotient manifold: $Gr(k, n) \cong O(n)/O(k) \times O(n-k)$. Similarly, we can also view the flag manifold $Fl(n_1, n_2, \cdots, n_d)$ as a quotient manifold of $O(n)$:

$$Fl(n_1, n_2, \cdots, n_d) \cong O(n)/(O(n_1) \times O(n_2) \times \cdots \times O(n_d))$$

where $\sum_{i=1}^{d} n_i = n$.

One thing to note is that the orthogonal group $O(n)$ has two disconnected components, i.e., n-by-n orthogonal matrices with determinant 1 and -1. Since our main purpose here is to develop algorithm based on geodesic and corresponding distance metric, we can constrain our computation to the special orthogonal group $SO(n)$, i.e. the set of n-by-n orthogonal matrices with determinant 1. Hence for computational purpose, we view the flag manifold as,

$$Fl(n_1, n_2, \cdots, n_d) \cong SO(n)/S(O(n_1) \times O(n_2) \times \cdots \times O(n_d)).$$

Also, $SO(n)/S(O(n_1) \times O(n_2) \times \cdots \times O(n_d))$ is referred to as the non-oriented flag manifold, which will be discussed further with partially and fully oriented flag in Section 5.4.

Now we can utilize this quotient structure to represent flags as matrices. Let $Q \in SO(n)$ be an n-by-n orthogonal matrix with positive determinant, the equivalence class $[Q]$ which represents a flag on $Fl(n_1, n_2, \cdots, n_d)$, is the set of orthogonal matrices,

$$[Q] = \left\{ Q \begin{pmatrix} P_1 & 0 & \cdots & 0 \\ 0 & P_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & P_d \end{pmatrix} : P_i \in O(n_i)\,,\ n_1 + n_2 + \cdots + n_d = n \right\}$$

and

$$Det \begin{pmatrix} P_1 & 0 & \cdots & 0 \\ 0 & P_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & P_d \end{pmatrix} = 1.$$

Follow the derivation shown in Section 2.2.1, it is straightforward to compute the vertical and horizontal space to a flag manifold of type $\mathbf{p} = \{n_1, n_2, \cdots, n_d\}$ as a quotient manifold of $O(n)$.

Recall that the vertical space is defined to be vectors tangent to the set $[Q] \in Fl(n_1, n_2, \cdots, n_d)$. One can compute the vertical space at Q, V_Q is the set of matrices of the form,

$$V_Q = Q \begin{bmatrix} A_1 & 0 & \cdots & 0 \\ 0 & A_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & A_d \end{bmatrix}$$

where $A_i \in \mathbb{R}^{n_i \times n_i}$ and $A_i = -A_i^T$. One may compute the orthogonal complement of V_Q in $T_Q O(n)$ with respect to the Euclidean metric. It follows that the horizontal space H_Q is the set of matrices of the form

$$H_Q = Q \begin{bmatrix} \mathbf{0}_{n_1} & & & * \\ & \mathbf{0}_{n_2} & & \\ & & \ddots & \\ -*^T & & & \mathbf{0}_{n_d} \end{bmatrix}$$

where $\mathbf{0}_{n_i}$ is an n_i-by-n_i zero matrix. The notation $*$ and $-*^T$ is to indicate the skew-symmetric structure. Matrices of this form are orthogonal to the vertical space V_Q with respect to the Euclidean metric. As discussed in Section 2.2.1, horizontal space provides a representation of tangent vectors to the quotient manifold, i.e. $Fl(n_1, n_2, \cdots, n_d)$. Hence the tangent space to a flag manifold of type $\mathbf{p}$ at a point $[Q]$, $T_{[Q]}Fl(n_1, n_2, \cdots, n_d)$ is given by the horizontal space H_Q. Thanks to the quotient structure of the flag manifold, to obtain the geodesic formula we just need to further constrain the velocity vector H in the orthogonal group geodesic

$$\phi(t) = Q \exp(tH)$$

to be living in the horizontal space H_Q [38]. Therefore the flag geodesic formula emanating from Q with velocity $\tilde{C}$ is

$$Q(t) = Q\exp(t\tilde{C}) \tag{5.1}$$

where $\tilde{C}$ is any skew-symmetric matrices of the form

$$\tilde{\mathbf{C}} = \begin{pmatrix} \mathbf{0}_{n_1} & & & * \\ & \mathbf{0}_{n_2} & & \\ & & \ddots & \\ -*^T & & & \mathbf{0}_{n_d} \end{pmatrix}, \ \mathbf{0}_{n_i} = \mathbf{0}^{n_i \times n_i}.$$

Since $\tilde{C}$ is a skew-symmetric matrix, one can compute the Youla decomposition [51] $\tilde{C} = U\Sigma U^T$ where U is an n-by-n orthogonal matrix and Σ is a block-diagonal matrix has the following form,

$$\Sigma = \begin{bmatrix} \begin{matrix} 0 & \lambda_1 \\ -\lambda_1 & 0 \end{matrix} & 0 & \cdots & 0 & & & \\ 0 & \begin{matrix} 0 & \lambda_2 \\ -\lambda_2 & 0 \end{matrix} & & 0 & & & \\ \vdots & & \ddots & \vdots & & & \\ 0 & 0 & \cdots & \begin{matrix} 0 & \lambda_r \\ -\lambda_r & 0 \end{matrix} & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{bmatrix}. \tag{5.2}$$

where $\pm\lambda_j i$ are the pure imaginary eigenvalues of $\tilde{C}$. This decomposition is also referred to as the spectral decomposition of skew-symmetric matrix and numerically this can be considered as a rearrangement of SVD of $\tilde{C}$. The essence of this factorization is that it provides a numerical formula to compute the exponential map, which in our case it is simply the matrix exponential. The flag geodesic

$$Q(t) = Q\exp(t\tilde{C})$$

can now be written as

$$Q(t) = Q\exp(Ut\Sigma U^T).$$

Expanding the right-hand side via Taylor's expansion yields

$$\begin{aligned} Q(t) &= QU\exp(t\Sigma)U^T && (5.3)\\ &= QUR(t)U^T && (5.4) \end{aligned}$$

where $R(t)$ is a block-diagonal matrix of the following form,

$$\begin{bmatrix} \begin{matrix}\cos(t\lambda_1) & -\sin(t\lambda_1)\\ \sin(t\lambda_1) & \cos(t\lambda_1)\end{matrix} & 0 & \cdots & 0 & & & \\ 0 & \begin{matrix}\cos(t\lambda_2) & -\sin(t\lambda_2)\\ \sin(t\lambda_2) & \cos(t\lambda_2)\end{matrix} & & 0 & & & \\ \vdots & & \ddots & \vdots & & & \\ 0 & 0 & \cdots & \begin{matrix}\cos(t\lambda_r) & -\sin(t\lambda_r)\\ \sin(t\lambda_r) & \cos(t\lambda_r)\end{matrix} & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix}. \quad (5.5)$$

Equation (5.4) also provides an intrinsic distance formula between $[Q(0)]$ and $[Q(1)]$

$$d([Q(0)],[Q(1)]) = \sqrt{\sum_{i=1}^{n} \lambda_i^2}$$

This concludes the numerical geodesic formula on the flag manifold given initial position and velocity, with which one can implement the steepest descent algorithm on the flag manifold. And yet we are also interested in the inverse operation: logarithmic map, i.e. given two points on a flag manifold, find the velocity vector and the corresponding distance.

5.2.2 Logarithmic map

In this section, we utilize equation (5.1) to find a numerical approximation of the geodesic between two points on a flag manifold. Let $[Q_1], [Q_2] \in Fl(n_1, n_2, \cdots, n_d)$ be two points on a flag manifold with matrix representation $Q_1, Q_2 \in SO(n)$, our goal is to find the solution to the following factorization

$$Q_2 = Q_1 \exp(H) M \tag{5.6}$$

for H and M. Here H and M are constrained to be of the form

$$H = \begin{pmatrix} \mathbf{0}_{n_1} & & & * \\ & \mathbf{0}_{n_2} & & \\ & & \ddots & \\ -*^T & & & \mathbf{0}_{n_d} \end{pmatrix} \text{ and } M = \begin{pmatrix} M_1 & 0 & \cdots & 0 \\ 0 & M_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & M_d \end{pmatrix}.$$

where H is skew-symmetric, $M_i \in O(n_i)$, $M \in SO(n)$. One may interpret equation (5.1) in the following way. In practice, two data points $[Q_1], [Q_2] \in Fl(n_1, n_2, \cdots, n_d)$ will be represented as two orthogonal matrix $Q_1, Q_2 \in SO(n)$. The geodesic velocity vector between $[Q_1]$ and $[Q_2]$ does not necessarily send Q_1 to Q_2. As illustrated in Figure 5.3, one may first map Q_1 to a representative in $[Q_2]$ via the exponential map with velocity H. Then this element in $[Q_2]$ is mapped to Q_2 via right multiplication of matrix M. For $Fl(k, n-k)$, i.e. the Grassmann manifold $Gr(k, n)$, one can solve for velocity matrix H analytically. See [19] for more details. For the more general flag manifold case, we will present an algorithm to find the numerical approximation of H and M in Section 5.3. Towards the goal of introducing the iterative algorithm, we may simplify

Equation (5.1) further by moving Q_1 to the identity(matrix), which can be done by multiplying Q_1^T on both sides of Equation (5.6). Let $Q = Q_1^T Q_2$, one can rewrite (5.6) as

$$Q = \exp(H)M. \tag{5.7}$$

To facilitate the description of the algorithm, a couple of notation needs to be introduced. We define $\mathcal{W}$as the vector space of all n-by-n skew-symmetric matrices. Set $\mathbf{p} = (n_1, n_2, \cdots, n_d)$. We define $\mathcal{W}_{\mathbf{p}}$ as the set of all block-diagonal skew-symmetric matrices of type $\mathbf{p}$, i.e.

$$\mathcal{W}_{\mathbf{p}} = \{G \in \mathcal{W} \mid G = \begin{pmatrix} G_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & G_d \end{pmatrix}\} \tag{5.8}$$

where $G_i \in \mathbb{R}^{n_i \times n_i}$ and $G_i = -G_i^T$. And correspondingly we can define its orthogonal complement in $\mathcal{W}$, i.e.

$$\mathcal{W}_{\mathbf{p}}^{\perp} = \{H \in \mathcal{W} \mid H = \begin{pmatrix} \mathbf{0}_{n_1} & & * \\ & \ddots & \\ -*^T & & \mathbf{0}_{n_d} \end{pmatrix}\} \tag{5.9}$$

where $\mathbf{0}_{n_i}$ is a n_i-by-n_i zero matrix.

Instead of solving Equation (5.7) for H and M directly, we propose to solve the following system of equations:

$$Q = \exp(H)\exp(G) \tag{5.10}$$

for H and G where $H \in \mathcal{W}_{\mathbf{p}}^{\perp}$ and $G \in \mathcal{W}_{\mathbf{p}}$. It is important to note that solving Equation (5.10) implicitly put our computation on, as we will call it, the fully-oriented flag manifold $SO(n)/(SO(n_1) \times SO(n_2) \times \cdots \times SO(n_d))$, which will be further discussed in Section 5.4. There is a 2^{d-1} to 1 map

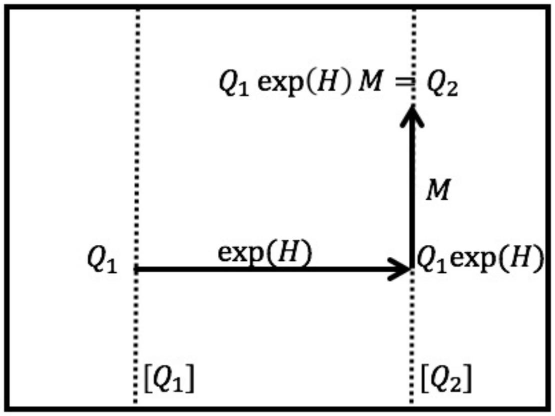


Figure 5.3: Illustration of Equation (5.7). The vertical lines represent the equivalence classes $[Q_1]$ and $[Q_2]$ respectively. Q_1 is mapped to an element in $[Q_2]$ by right multiplication with $\exp(H)$ which is then sent to Q_2 by multiplying with M.

from the fully-oriented flag manifold to the flag manifold(non-oriented flag manifold). To find H and distance between two flags $[Q_1]$ and $[Q_2]$, one needs to find the optimal H with the shortest distance arise from the output of the algorithm.

5.3 Numerical Algorithm for Log map

Recall that the equation we want to solve is

$$Q = \exp(H)\exp(G)$$

with constraints $H \in \mathcal{W}_{\mathbf{p}}^{\perp}$ and $G \in \mathcal{W}_{\mathbf{p}}$. $\mathcal{W}_{\mathbf{p}}$ and $\mathcal{W}_{\mathbf{p}}^{\perp}$ are defined in Equation (5.8) and (5.9). The idea of our Iterative Alternating algorithm is straightforward. First we define

$$\text{Proj}_H : \mathbb{R}^{n\times n} \mapsto \mathcal{W}_{\mathbf{p}}^{\perp}$$

as a projection of n-by-n matrices to the set of n-by-n skew-symmetric matrices in $\mathcal{W}_{\mathbf{p}}^{\perp}$, and also

$$\text{Proj}_G : \mathbb{R}^{n\times n} \mapsto \mathcal{W}_{\mathbf{p}}$$

$$G^{(0)} \longrightarrow \hat{H} = Log\left(Q \exp\left(G^{(0)}\right)^{T}\right) \longrightarrow H^{(1)} = Proj_{H}(\hat{H})$$

$$H^{(1)} \longrightarrow \hat{G} = Log\left(\exp\left(H^{(1)}\right)^{T} Q\right) \longrightarrow G^{(1)} = Proj_{G}(\hat{G})$$

......

Iterating until convergence

Figure 5.4: Demonstration of the iterative alternating algorithm

as a projection to the set of n-by-n block-diagonal skew-symmetric matrices of type $\mathbf{p}$ in $\mathcal{W}_{\mathbf{p}}$. As illustrated in Figure 5.4, given an initial guess $G^{(0)} \in \mathcal{W}_{\mathbf{p}}$, one can "solve" for H numerically. Let $\hat{H} = \log(Q \cdot \exp(G^{(0)})^T)$, since $\hat{H}$ is in general not living in $\mathcal{W}_{\mathbf{p}}^{\perp}$), we project $\hat{H}$ onto $\mathcal{W}_{\mathbf{p}}^{\perp}$ to obtain the updated H. This projection simply sets certain select entries in $\hat{H}$ to be zero, which is denoted by $H^{(1)} = \mathrm{Proj}_{\mathcal{W}_{\mathbf{p}}^{\perp}}(\hat{H})$. Then we turn to solve for G in the same way. Let $\hat{G} = \log(\exp(H^{(1)})^T Q)$ we project $\hat{G}$ onto $\mathcal{W}_{\mathbf{p}}$, i.e. $G^{(1)} = \mathrm{Proj}_G(\hat{G})$. Then iterate this process until it converges. The pseudo-code of our Iterative Alternating algorithm is presented in Algorithm 7.

It is important to note in this section, $\exp$ and $\log$ are used to denote matrix exponential and principal matrix logarithm. Please refer to [2, 3] for details about computing principal matrix logarithm. Since we are doing computation on matrices from $SO(n)$, it is straightforward to show the eigenvalues of the special orthogonal matrix are unit complex numbers. This indicates that the principal matrix logarithm is defined as long as -1 is not an eigenvalue of a special orthogonal matrix, which we never observed in our computation.

Recall that the original equation we want to solve is $Q = \exp(H) \cdot M$, while our iterative algorithm is working with $Q = \exp(H)\exp(G)$. Since G is a block-diagonal skew-symmetric matrix of type $\mathbf{p}$, the image of the exponential map $\exp(G)$ is a block-diagonal special orthogonal matrix, i.e. each block along the diagonal is a n_i-by-n_i special orthogonal matrix. Our algorithm is working on the fully-oriented flag manifold $SO(n)/(SO(n_1) \times SO(n_2) \times \cdots \times SO(n_d))$ instead of the non-oriented flag manifold of type $\mathbf{p}$, $SO(n)/S(O(n_1) \times O(n_2) \times \cdots \times O(n_d))$. As we will see in the next Section, a fully-oriented flag manifold of type $\mathbf{p} = (n_1, n_2, \cdots, n_d)$ is a 2^{d-1} cover to the corresponding flag manifold of type $\mathbf{p}$.

5.4 Partially and fully oriented flag manifold

In this section, we will introduce the fully-oriented flag manifold which we encounter during the implementation of our Iterative Alternating algorithm. To the best of our knowledge, fully-oriented flag manifold is barely discussed in the literature under the context of data analysis. In [42], oriented flag manifold is used to describe the possible positions of a rigid body. As previously mentioned, the fully-oriented flag manifold is defined as a quotient manifold of $SO(n)$. Let $P \in SO(n)$ be an n-by-n special orthogonal matrix. The equivalence class $[P]$ which represents a point on the fully-oriented flag manifold of type $\mathbf{p} = (n_1, n_2, \cdots, n_d)$ is the set of orthogonal matrices,

$$[P] = \left\{ P \begin{pmatrix} P_1 & 0 & \cdots & 0 \\ 0 & P_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & & P_d \end{pmatrix} : P_i \in SO(n_i)\,,\, n_1 + n_2 + \cdots + n_d = n \right\},$$

We denote the fully-oriented flag manifold as $Fl^+(n_1, n_2, \cdots, n_d)$.

Recall that when we define the equivalence class in non-oriented flag manifold, the diagonal elements P_i's are only required to be living in $O(n_i)$ and $diag\{P_1, P_2, \cdots, P_d\}$ as a whole should be living in $SO(n)$. For the fully-oriented flag, P_i's are required to be n_i-by-n_i special orthogonal matrices. That is to say, we now take the orientation of each subspace in the flag structure into consideration. For example, the non-oriented flag $Fl(1,1,1)$ is the set of a 1-dimensional line in a 2-dimensional plane in the 3-dimensional space. The corresponding fully-oriented flag $Fl^+(1,1,1)$ is the set of directed 1-dimensional line in a directed 2-dimensional plane living in the $\mathbb{R}^3$ space. One can verify that the fully-oriented flag $Fl^+(n_1, n_2, \cdots, n_d)$ is a 2^{d-1} cover to the corresponding non-oriented flag manifold $Fl(n_1, n_2, \cdots, n_d)$.

To view the partially-oriented flag manifold, we need to decorate the notation we introduced above a bit. We will explore the fully-oriented and partially-oriented flag manifold via a matrix example. Let $Fl(2,2,2) = SO(6)/S(O(2) \times O(2) \times O(2))$ be a non-oriented flag manifold. Denote the first $O(2)$ by a, the second $O(2)$ by b and the third $O(2)$ by c. The non-oriented flag $Fl(2,2,2)$ is denoted by $Fl_{abc}(2,2,2)$ where subscript abc is used to emphasize that each $O(2)$ component can have positive or negative unit determinant but together, the whole matrix has determinant 1, i.e., $S(O(2) \times O(2) \times O(2))$ is the subgroup of block-diagonal matrices

$$\begin{bmatrix} P_1 & & \\ & P_2 & \\ & & P_3 \end{bmatrix}, P_i \in O(2), \prod_{i=1}^{3} det(a_i) = 1.$$

Then the corresponding fully-oriented flag can be denoted as $Fl_{a,b,c}(2,2,2) = \mathrm{SO}(6)/\mathrm{SO}(2) \times \mathrm{SO}(2) \times \mathrm{SO}(2)$ where this time the comma-separated subscript a, b, c is used to emphasize that each component along the diagonal has unit determinant, i.e., $SO(2) \times SO(2) \times SO(2)$ is the subgroup of block-diagonal matrices

$$\begin{bmatrix} Q_1 & & \\ & Q_2 & \\ & & Q_3 \end{bmatrix}, Q_i \in SO(2).$$

Follow this subscript notation, $Fl_{ab,c}(2,2,2) = SO(6)/S(O(2) \times O(2)) \times O(2)$ is one of the three associated partially-oriented flag manifolds. The diagram which describes the relations between non-oriented flag, partially oriented flag and fully-oriented flag is demonstrated in Figure 5.5.

Let $A_1 \in \mathrm{SO}(6)$ be a 6-by-6 orthogonal matrix with determinant 1, also let

$$A_2 = A_1 \begin{bmatrix} -1 & & & & & \\ & 1 & & & & \\ & & -1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{bmatrix}, A_3 = A_1 \begin{bmatrix} -1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & -1 & \\ & & & & & 1 \end{bmatrix}$$

$$A_4 = A_1 \begin{bmatrix} 1 & & & & & \\ & 1 & & & & \\ & & -1 & & & \\ & & & 1 & & \\ & & & & -1 & \\ & & & & & 1 \end{bmatrix}.$$

A_1, A_2, A_3, A_4 are four different points on the fully-oriented flag $Fl_{a,b,c}(2,2,2)$. However, they are living in the same equivalence class on the non-oriented flag $Fl_{abc}(2,2,2)$. A_1 and A_2 are different representations of the same point on partially-oriented flag manifold $Fl_{ab,c}(2,2,2)$. Similarly, A_1 and A_3 represent the same point on $Fl_{ac,b}(2,2,2)$, also A_1 and A_4 represent the same point on $Fl_{a,bc}(2,2,2)$.

As discussed in Section 5.3, our Iterative Alternating algorithm is implicitly implemented on the fully-oriented flag manifold. Given a non-oriented flag $[Q] \in Fl(n_1, n_, 2, \cdots, n_d)$, there are 2^{d-1} different representations of $[Q]$ on the corresponding fully-oriented flag manifold. Since our goal is to compute distance and geodesic on the non-oriented flag, we need to pass all 2^{d-1} representations, $\{Q_i\}_{i=1}^{2^{d-1}}$,of $[Q]$ to the Iterative Alternating algorithm and solve $Q_i = \exp(H_i)\exp(G_i)$, $i = 1, \cdots, 2^{d-1}$ for the optimal H which gives the shortest distance. Recall that the distance associated with H is

$$d = \sqrt{\sum_{k=1}^{l} \lambda_k^2}$$

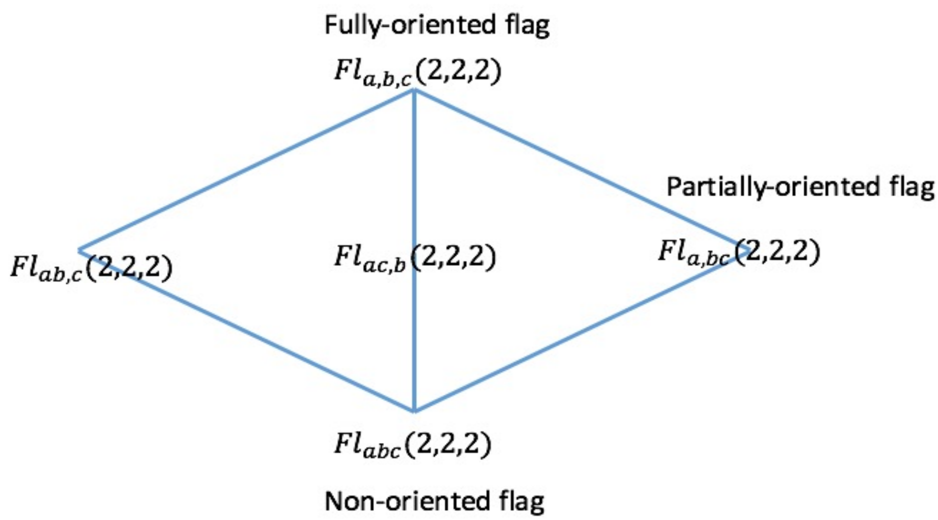


Figure 5.5: This diagram shows the relations between non-oriented flag, partially oriented flag and fully oriented flag.

where $\pm i\lambda_k$ are eigenvalues of H. Here we will modify the Iterative Alternating algorithm introduced in Section 5.3 so that it covers all representations on the corresponding fully-oriented flag to output the optimal velocity matrix H to recover the geodesic on the non-oriented flag manifold. The pseudo-code for Iterative Alternating Algorithm is presented in Algorithm 7 and the accessory pseudo-code that generates all representations of $[Q] \in Fl(n_1, n_2, \cdots, n_d)$ is also presented in Algorithm 8. An overview of the main algorithm is presented as follows,

1. Present two (special) orthogonal matrix representations (of data sets) $X_1, X_2 \in SO(n)$ and the flag structure $\mathbf{p} = \{n_1, \cdots, n_d\}$ to the algorithm. Move X_1 to the origin (identity): $Q = X_2^T X_1$.

2. Compute all 2^{d-1} elements of Q in the fully oriented manifold via Algorithm 8: $\{Q_i\}_{i=1}^{2^{d-1}}$ = generateQi(Q,**p**)

3. For each element $Q_i \in \{Q_i\}_{i=1}^{2^{d-1}}$, solve Equation (5.10) using Algorithm 7: $H_i^{(j)}, G_i^{(j)}$ = iterativeSolver(Q_i,**p**), iterate this process M times, i.e. $j = 1, \cdots, M$. Find the solution as-

sociated with the minimum distance: $H_i^* = \arg\min \sqrt{\frac{1}{2}\mathrm{Tr}(H_i^{(j)T} H_i^{(j)})}$ to obtain the *shortest* geodesic (on the corresponding partially oriented flag).

4. Among all the *shortest* geodesics on partially oriented flags, find the *shortest* geodesic on the fully oriented flag: $H^* = \arg\min \sqrt{\frac{1}{2}\mathrm{Tr}(H_i^{*T} H_i^*)}$

The pseudo code for the main algorithm is presented in Algorithm 6 calling subroutine Algorithm 7 and Algorithm 8.

Algorithm 6: Main algorithm

Input Data: $X_1, X_2 \in SO(n)$, $\mathbf{p} = (n_1, n_2, \ldots, n_d)$,M,maxIter,$\epsilon$
Output Data: H^*, G^*
Define: d(H) = $\sqrt{\dfrac{1}{2}\mathrm{Tr}(\mathrm{H^T H})}$

Function `main(`X_1, X_2, $\mathbf{p}$`)`**:**
    $\mathrm{Q} = \mathrm{X_1}^T \mathrm{X_2}$
    $\mathrm{d}^* = \infty$
    $\{\mathrm{Q_i}\}_{\mathrm{i=1}}^{2^{\mathrm{d-1}}}$ = generateQi(Q,**p**)
    for Q *in* $\{\mathrm{Q_i}\}_{\mathrm{i=1}}^{2^{(\mathrm{d-1})}}$ **do**
        for $\mathrm{i} = 1, \cdots, \mathrm{M}$ **do**
            H, G = iterativeSolver(Q,**p**,maxIter,ϵ)
            if $\mathrm{d}^* > \mathrm{d(H)}$ **then**
                d^*, H^*, G^* = d(H), H, G
        end
    end
    return $\mathrm{d}^*, \mathrm{H}^*$,$\mathrm{G}^*$

5.5 2k Embedding

For many practical applications, the trailing n_d columns are not of interest, e.g. computations on $Fl(k, n-k) = Gr(k, n)$ are usually performed using n-by-k orthonormal matrices since only the first k columns are of interest. Here in this section we will prove that the iterative algorithm 7 can be performed in a lower dimensional space if $k = \sum_{i=1}^{d-1} n_i$ is relatively small, more specifically, if $k < n/2$.

Algorithm 7: Iterative Alternating algorithm

```
Input Data: Q ∈ SO(n), p = (n_1, n_2, ..., n_d), maxIter, ϵ
Output Data: H^(k), G^(k)
Function iterativeSolver(Q, p),maxIter,ϵ:
    Generate random G^(0)
    k = 0
    while k ≤ iterMax and err < ϵ do
        k = k + 1
        H^(k) = P_H(log(Q exp(−G^(k−1))))
        G^(k) = P_G(log(exp(−H^(k))Q))
        err = ‖Q − exp(H) exp(G)‖_F
    end
    return H^(k),G^(k)
```

Algorithm 8: Fully-oriented flag representations(MATLAB pseudo code)

```
Function generateQi(Q,p):
    colHeader = [0,cumsum(p)]+1
    m = length(colHeader)
    n = floor(d/2)
    i = 1
    Q_i = Q
    for j = 1 : n do
        C = nchoosek(colHeader, 2*j)
        for k = 1: size(C,1) do
            i = i + 1
            Q_i = Q
            Q_i(:, C(k,:)) = -Q_i(:, C(k,:))
        end
    end
    return {Q_i}_{i=1}^{2^(d−1)}
```

Without loss of generality, the geodesic between two flags of type $\mathbf{p} = (n_1, n_2, \cdots, n_d)$ can always be identified with a geodesic between the identity matrix, I, and some $Q \in SO(n)$ by moving the initial point to I, i.e.,

$$Q = I \exp(\begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}) \tag{5.11}$$

where $k = \sum_{i=1}^{d-1} n_i$, $B \in \mathbb{R}^{(n-k)\times k}$ and A is a k-by-k skew-symmetric matrix of the form

$$A = \begin{bmatrix} \mathbf{0}_{n_1} & -B_{2,1}^T & \cdots & -B_{d-1,1}^T \\ B_{2,1} & \mathbf{0}_{n_2} & & -B_{d-1,2}^T \\ \vdots & & \ddots & \vdots \\ B_{d-1,1} & B_{d-1,2} & \cdots & \mathbf{0}_{n_{d-1}} \end{bmatrix}. \tag{5.12}$$

$Q(t) = I \exp(t \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}), t \in [0, 1]$ traces an n-by-n representation of the geodesic flow between $[I]$ and $[Q]$. The following theorem and its corollary provides a method to perform the iterative algorithm 7 with $2k$-by-$2k$ matrices instead of n-by-n matrices.

Theorem 2. *Let* $[Q] \in Fl(n_1, n_2, \cdots, n_d)$. *Suppose* $Q(t) = \exp(t \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix})$ *with* $Q(0) = I$, $Q(1) = Q$ *is a flag geodesic flow between* $[I]$ *and* $[Q]$. *If*

$$q(t) = \exp(t \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix}) I_{n,k} \tag{5.13}$$

and $\mathrm{span}\{q(0)\} \cap \mathrm{span}\{q(1)\} = \{0\}$, *then for all* $t \in [0, 1]$, $\mathrm{span}\{q(t)\} \subset \mathrm{span}\{[q(0), q(1)]\}$, *where* $k = \sum_{i=1}^{d-1} n_i$ *and* $I_{n,k}$ *denotes the first* k *columns of an* n*-by-*n *identity matrix.*

Note that if $2k \geq n$, Theorem 2 is trivial. So here we assume $2k < n$. Before proving the theorem, we need to introduce some notation. Let $q := QI_{n,k} = q(1)$ be the first k columns of Q.

In fact, $q(t)$ defined in Equation (5.13) can be understood as a geodesic path between $I_{n,k}$ and q by viewing $Fl(n_1, n_2, \cdots, n_d)$ as a quotient manifold of the Stiefel manifold $St(k, n)$ (refer to [50] for more details). Further, we write the n-by-k orthonormal matrix q in block matrix form as

$$q = \begin{bmatrix} q_k \\ q_{n-k} \end{bmatrix} \tag{5.14}$$

where q_k and q_{n-k} denote the first k rows and the trailing $n-k$ rows of q respectively.

Lemma 5.5.1. *If $q(t)$ is defined as in Equation (5.13), such that $q(0) = I_{n,k}$ and $q(1) = q$, then* $\operatorname{span}\{q_{n-k}\} = \operatorname{span}\{B\}$.

Proof. Let $U_B R_B := B$ be the compact QR decomposition of B (U_B: $(n-k)$-by-k, R_B: k-by-k). Define

$$f(t) = (I - U_B U_B^T) J q(t) \tag{5.15}$$

where $J = \begin{bmatrix} 0 & I_{n-k} \end{bmatrix}$ is the last $n-k$ rows of the n-by-n identity matrix. Hence left multiplication by J on $q(t)$ simply selects the last $n-k$ rows of $q(t)$. By definition $f(0) = 0$. Differentiate $f(t)$ to get:

$$\dot{f}(t) = (I - U_B U_B^T) J \begin{bmatrix} A & -B^T \\ B & 0 \end{bmatrix} q(t) = 0 \tag{5.16}$$

Therefore, $f(t) \equiv 0$ for $t \in [0, 1]$. If we evaluate $f(t)$ at $t = 1$, we get:

$$f(1) = (I - U_B U_B^T) q_{n-k} = 0 \tag{5.17}$$

By the assumption that $q(0)$ and $q(1)$ do not intersect, we know q_{n-k} is of rank k hence U_B is also of rank k. The conclusion follows. □

Now we present a proof to the theorem.

Proof. Let $UR := [I_{n,k}, q]$ be the thin QR-decomposition of $[q(0), q(1)]$. Consequently, U is an orthonormal basis for $\mathrm{span}\{[q(0), q(1)]\}$. The n-by-k orthonormal matrix U takes the block form

$$U = \begin{bmatrix} I_k & 0 \\ 0 & C \end{bmatrix}. \tag{5.18}$$

Note that $\mathrm{span}\{C\} = \mathrm{span}\{q_{n-k}\}$ where q_{n-k} is defined in Equation (5.14). Define

$$g(t) = (I - UU^T)q(t). \tag{5.19}$$

By definition, $g(0) = (I - UU^T)I_{n-k} = 0$. If we differentiate $g(t)$, we get:

$$\dot{g}(t) = \begin{bmatrix} 0 & 0 \\ (I_{n-k} - CC^T)B & 0 \end{bmatrix} q(t) \tag{5.20}$$

By Lemma 5.5.1, $\mathrm{span}\{B\} = \mathrm{span}\{q_{n-k}\} = \mathrm{span}\{C\}$. We conclude that $\dot{g}(t) \equiv 0$, which implies $g(t) \equiv 0$. Therefore $q(t)$ is always living in the span of $[q(0), q(1)]$. $\square$

The theorem shows that the flag geodesic flow $q(t)$ between $I_{n,k}$ and q never leaves the $2k$-dimensional subspace $\mathrm{span}\{[I_{n,k}, q]\}$, which leads to the conclusion that the logarithmic map computation can be performed within this $2k$ dimensional space without loss of information. Here we introduce the following corollary.

Corollary 1. *Suppose $q(t)$ is defined as in Equation (5.13) such that $q(0) = I_{n,k}$ and $q(1) = q$. Let $UR := [I_{n,k}, q]$ be the compact QR-decomposition of $[q(0), q(1)]$, then $\phi(t) = U^T q(t)$ is a geodesic flow between $\phi(0) = U^T q(0)$ and $\phi(1) = U^T q(1)$ on $Fl(n_1, n_2, \cdots, n_{d-1}, k)$. Moreover, $d(\phi(0), \phi(1)) = d(q(0), q(1))$ and $q(t) = UU^T \phi(t)$.*

This corollary can be proved by combining the results from Theorem 2 and Corollary 2.2 in [19].

5.6 Numerical Experiments

5.6.1 Ellipsoid data

The purpose of this synthetic example is to show the difference between flag geodesic and Grassmannian geodesic, as well as their corresponding geodesic distance under the context of comparing data sets. As can be seen in Figure 5.6, each ellipsoid data cloud contains 100 data points in $\mathbb{R}^3$. Let $\{r_i\}$ and $\{b_i\}$ denote the data points in the red and blue ellipsoid respectively. Each data set can be written as a short wide data matrix $[r_1, r_2, \cdots, r_{100}] = R \in \mathbb{R}^{3\times 100}$ and $[b_1, b_2, \cdots, b_{100}] = B \in \mathbb{R}^{3\times 100}$. We denote the SVD basis for each ellipsoid data set by $U_R = [u_R^{(1)}, u_R^{(2)}, u_R^{(3)}]$ and $U_B = [u_B^{(1)}, u_B^{(2)}, u_B^{(3)}]$. One can view the SVD basis as giving the major, medium, and minor axes of the corresponding ellipsoid.

The Grassmannian geodesic distance between two bases is 0 since the columns of U_R or U_B span all of $\mathbb{R}^3$. To compare two ellipsoids via the Grassmannian setting, one would typically represent the data sets with their first principal components namely $u_R^{(1)}$ and $u_B^{(1)}$, and then compute the distance between these two vectors on $Gr(1, 3)$. Hence the Grassmannian geodesic between two ellipsoids is the path between two major axes and the distance is the angle between the major axes. The information contained in the relationship between the other two axes is lost. Note that this limitation comes from the Grassmannian rather than the data itself.

By representing two ellipsoids of data points by their SVD bases U_R, U_B such that $[U_R]$, $[U_B] \in Fl(1, 1, 1)$, one has finer resolution to describe the corresponding ellipsoids since $Fl(1, 1, 1)$ has dimension 3 (while $Gr(1, 3)$ has dimension 2). The geodesic between two flag representations correspondingly encodes more information than moving one major axis to another in the Grassmannian setting.

5.6.2 MNIST image data set

Here we utilize the well-studied MNIST data set to illustrate the use of the flag manifold for comparing sets of SVD bases of "mixed" digits. We select hand written digits "1" and "5" from the training set of the MNIST data set, where each digit is a 28×28 image. All images are vectorized

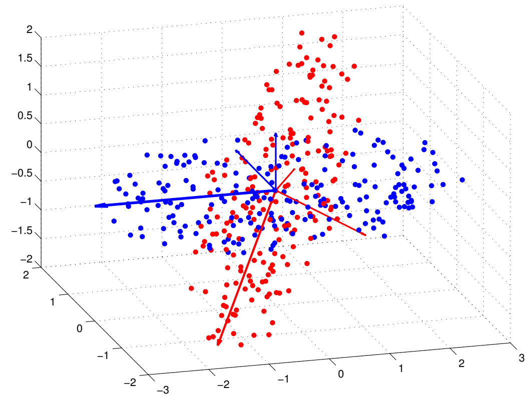
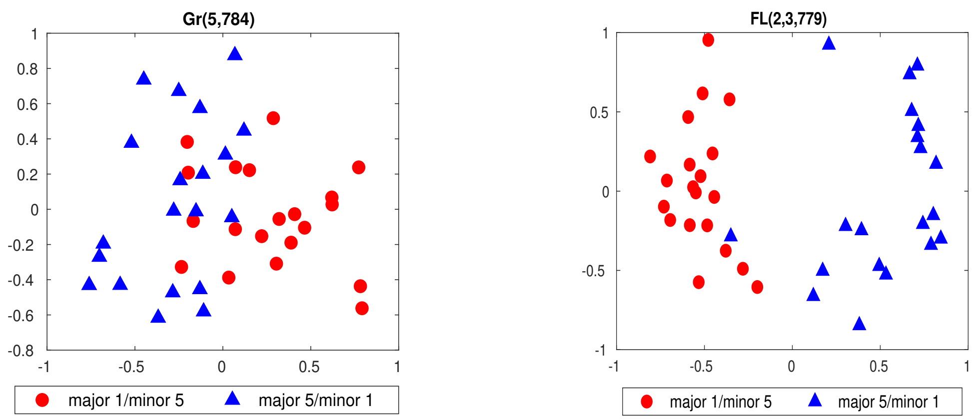


Figure 5.6: Two sets of ellipsoid shaped data points in $\mathbb{R}^3$. Each SVD basis can be viewed as a point on $Fl(1,1,1)$

.

and centered by subtracting the mean of all images. Then we form a set of mixed digits data sets consisting of two classes, namely "major $1/$minor 5" and "major $5/$minor 1". "major $1/$minor 5" (resp. "major $5/$minor 1") is formed by concatenating m "1"'s (resp. m "5"'s) and p "5"'s (resp. p "1"'s). In general m is assumed to be larger then p. Hence each data set is represented by a $784 \times (m+p)$ matrix. We compute the SVD basis for each $784 \times (m+p)$ matrix and select the first k columns of the SVD basis as a representation for each data set. Thus each data set is represented by a $784 \times k$ orthonormal matrix. For the following experiment $m = 16$, $p = 9$ and $k = 5$. We may consider each 784×5 SVD basis as a data point on $Fl(2,3,779)$ or $Gr(5,784)$. The first 5 eigen-digits for both of the two classes in this experiment are demonstrated in Figure 5.8 and Figure 5.9. One can compute the pairwise flag and Grassmannian geodesic distance to form the corresponding distance matrix. We then embed these data points to the Euclidean space by multi-dimensional scaling.

In Figure 5.7, we see the configurations of MDS using Grassmannian(5.7a) and flag distance(5.7b). We observe that in 5.7a, the Grassmannian MDS configuration is showing overlapping between two classes. This is not surprising since each data point, no matter which class, is capturing the span of "1"'s and "5"'s. As can be seen in 5.7b, there is a clear separation between two classes except for one point. Note the input matrices fed to the algorithm are identical for both configurations. The difference is purely coming from the effect of the flag structure.



(a) Grassmannian MDS configuration

(b) Flag MDS configuration

Figure 5.7: Comparison of Grassmannian and flag MDS configurations

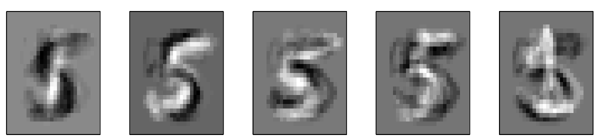


Figure 5.8: First 5 eigen digits of major 5/minor 1 data set

Figure 5.9: First 5 eigen digits of major 5/minor 1 data set

5.6.3 Indian Pines hyperspectral image data

To illustrate the utility of the proposed flag model in comparing real data sets, we apply it to the Indian Pines hyperspectral image data set. The hyperspectral images in this data set are

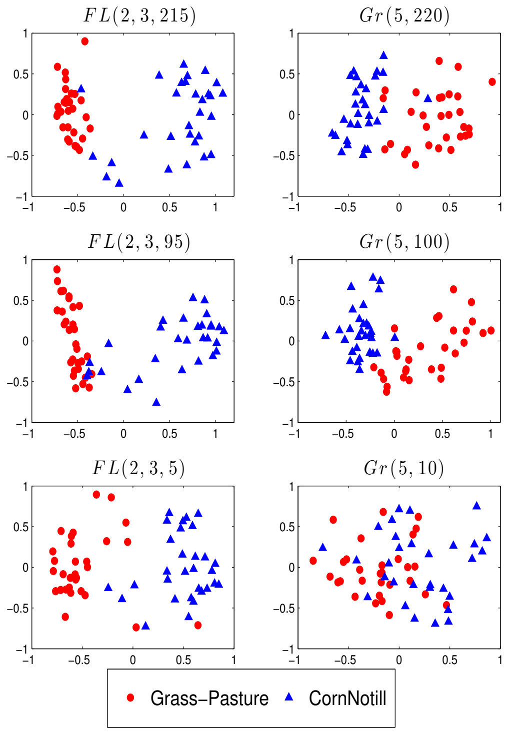


Figure 5.10: A comparison(horizontal) of the Grassmannian and Flag manifolds for representing data sets. The subspace dimension k fixed while the ambient dimension n is varying from 220,100 to 10.

145×145 pixels by 220 spectral bands (from $0.4\mu m$ to $2.4\mu m$). 10366 pixels are labelled and each is assigned to one of the 16 classes. Here we will test both the flag model and the Grassmann model on the task of visualizing sets of data sets.

For a chosen dimension k (note that $k = \sum_{i=1}^{d-1} n_i$ for $Fl(n_1, n_2, \cdots, n_d)$), we assemble 30 $n \times k$ matrices X_i from each class (so $p = 60$ data matrices total). Each data matrix consists of k 200×1 data vectors which belong to one of the two classes. Then for each matrix X_i, a compact SVD is applied to obtain an SVD/PCA basis, hence each data point (subspace) is represented by a $220 \times k$ orthonormal matrix U_i where $U_i \Sigma_i V_i^T = X_i$. The distance between SVD bases, assumed as representatives for points on a given flag manifold, can then be computed to obtain a $p \times p$ distance matrix. We use this distance matrix to embed these flags as points in Euclidean space via Multi-Dimensional Scaling (MDS). The first two coordinates of the optimal Euclidean configuration are selected for visualization in $\mathbb{R}^2$. Figure 5.10 illustrates the Euclidean embedding configurations for fixed subspace dimension $k = 5$ with various ambient dimensions using both the Grassmannian geodesic distance and flag distance. The ambient space is selected to be the n spectral bands with highest responses for $n = 100, 10, 5$. It is observed in the first two rows that both Grassmannian and flag geodesic distance provide a good separation with relatively large ambient dimension at $n = 220$ and 100. When the ambient dimension is reduced to $n = 10$, the third row of Figure 5.10 shows that the flag distance MDS embedding separates two classes in $\mathbb{R}^2$ while the Grassmannian MDS embedding shows heavy overlapping. Figure 5.11 shows the eigenvalues corresponding to the MDS embedding using flag distance on $Fl(2, 3, 5)$ (left) and $Gr(5, 10)$ (right). As we can see, the largest eigenvalue on the left panel is dominating which also suggests that flag MDS configurations are separable in lower dimension, which we don't observe in the Grassmannian MDS eigenvalues plot. Figure 5.12 shows, for fixed ambient dimension $n = 220$, how sets of data sets are pulled apart by increasing the dimension in the flag structure. From top left, we observe that the embedding of data points on $Fl(1, 219)$ to $\mathbb{R}^2$ live on a circle and are not separable. As we increase the flag structure dimension, the corresponding MDS configurations start to show more separation and for $Fl(1, 4, 215)$, the embedding of two classes is linearly separable.

In Figure 5.13, we select 6 bands (bands: 3,29,42,61,65,158) and use 20 pixels within the same class to form a data matrix of size 6×30. Each class consists of 20 such short and wide matrices and each matrix is represented by its 6-by-6 SVD basis and assumed to be representatives for points on $Fl(2, 2, 2)$. The pairwise distance is computed to obtain MDS configurations on $\mathbb{R}^2$. It is observed that the MDS embeddings of 3 classes are separable in low dimensional space with only 6 bands.

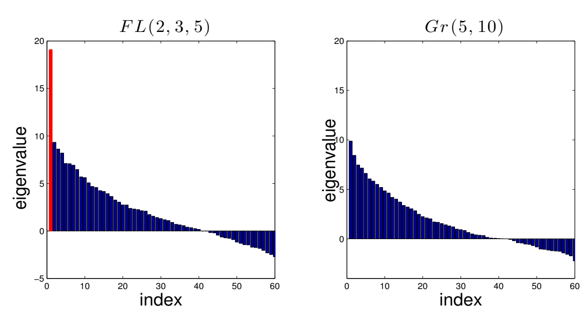


Figure 5.11: Eigenvalues of MDS for Left:$Fl(2, 3, 5)$, Right:$Gr(5, 10)$ in descending order.

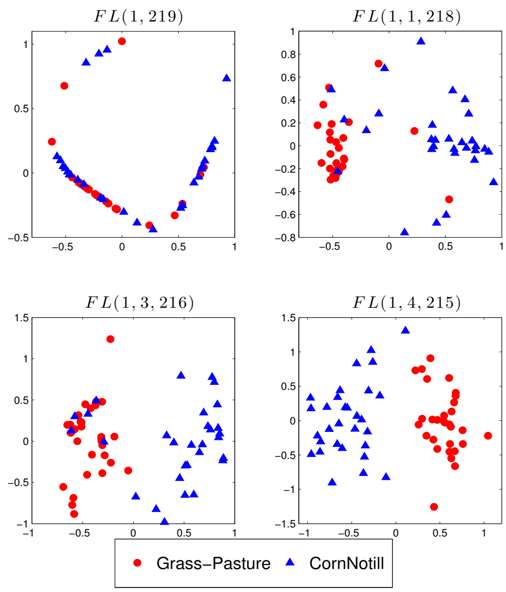


Figure 5.12: Configuration of points on various flag manifolds embedded in Euclidean space.

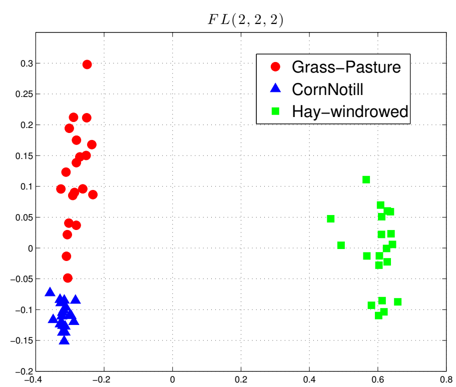


Figure 5.13: Configuration of points on $Fl(2,2,2)$ embedded in Euclidean space for 3 classes: Grass-Pasture',Corn-notill,Hay-windrowed. 6 bands(3,29,42,61,65,158) are selected so the ambient dimension $n = 6$.

5.7 Conclusion

In this chapter, we proposed a geometric framework for computing distance between nested subspaces, i.e. points on a flag manifold. This approach exploits the geometric feature of the flag manifold to help data analyst gain insight into how the data resides in the ambient space.

We have presented an overview of the theoretical foundation for computing distance between two flags. The numerical algorithm for computing the distance as well as set of points along the shortest path between flags follows the theory naturally. The "2k-embedding" trick along with the proof to it are introduced for practical utilization of the algorithm on real world data set. These tools allow one to analyze data sets using the intrinsic flag distance where the Grassmannian setting may fail.

The flag geodesic algorithm is demonstrated on the MNIST data sets where the classes in the experiment are mixed. The mixture causes confusion in the Grassmannian setting while the flag setting is still able to separate two mixed classes. This approach is also applied on the Indian Pines hyper-spectral image data sets. In this experiment, we focus on the transition from tall matrices to wide matrices. We observe that when the ambient dimension is close to the feature dimension, the flag geodesic distance is able to separate the data for visualization in 2-dimensional space while the Grassmannian fails to do so.

Chapter 6

Conclusion

6.1 Contribution

The focus of this dissertation is on developing numerical algorithms for comparing sets of data sets on the Grassmannian and flag manifold. To be specific, we made the following contribution.

- We rewrite the classical gappy Proper Orthogonal Decomposition algorithm as an optimization problem on the Grassmann manifold and solved it via steepest gradient descent. Having an objective function allows us to introduce penalty term which helps the algorithm to avoid local minimum. The algorithm is applied to the synthetic travelling sinusoidal wave data and facial image data.

- We show that by utilizing the affine Grassmannian framework, one can solve for the analytical solution to the gappy data imputation problem.

- We extend the self-organizing mappings to the settings of the Grassmann manifold by utilizing the geometric framework of the Grassmannians. We demonstrate this approach on the Indian Pines hyper-spectral data and influenza gene expression data for visualization of high dimensional data on 2-dimensional plane. We observe clear separation using the Grassmannian framework while the Euclidean space separation is known to be a difficult task.

- We introduce an iterative alternating algorithm to compute the logarithmic map on the flag manifold. The algorithm allows one to compute the distance as well as the set of points along the shortest path between two points on a flag manifold.

- We prove theoretical results to reduce the computational cost of the iterative algorithm.

- We propose the idea of analyzing sets of data sets using the geometric framework of the flag manifold. We demonstrate the algorithm on the "mixed" MNIST data set and Indian Pine hyper-spectral data set and observe that the flag framework separates data in 2-dimensional space while the Grassmannian fails to, especially when the ambient dimension is close to the feature dimension.

6.2 Future Work

There are several directions that this work can potentially be advanced in the future. In Chapter 3, we would like to further explore the geometric feature of the affine Grassmannian manifold. Although the affine Grassmannian setting leads to an analytical solution to the gappy data imputation problem, we yet find any real-world problem that exploits the affine Grassmannian framework. Can we build a better geometric understanding of affine Grassmannian under the context of missing data imputation problem? Can we find real-world problems that is better solved using the affine Grassmannian framework?

With respect to Chapter 4, the Grassmannian SOM algorithm requires every data points, i.e., linear subspaces to be of the same dimension. One might want to generalize the algorithm so that the data points are allowed to be of different dimensions. Subspaces of different dimensions are not living on a Grassmann manifold, but there are tools like Schubert varieties on can use to define shortest path and distance between subspaces of different dimensions.

For the iterative algorithm introduced in Chapter 5, we want to expand its application to the area of domain adaptation. The Grassmannian geodesic has been perceived as a geometric tool for conveying information between two domains. As a generalization and a refined version of the Grassmannians, can we utilize the flag geodesic to tackle the domain adaptation problem?

In Chapter 5, we have demonstrated the theoretical foundations as well as the numerical algorithm of the flag manifold. There are several interesting observations that could potentially be proved.

- In Figure 5.11, we observe that number of positive eigenvalues associated with the Grassmannian MDS and flag MDS are identical. Can we prove this observation theoretically?

- We observe that the smallest angle between $[Q_1], [Q_2] \in Gr(k, n)$ are identical to the smallest angle between $[Q_1], [Q_2] \in Fl(n_1, n_2, \cdots, n_d)$ where $\sum_{i=1}^{d} n_i = k$, i.e., the smallest singular value of the Grassmannian velocity matrix is equal to that of the flag velocity matrix between Q_1 and Q_2. Can we prove this result?

The theory and applications developed in this dissertation expand the toolbox for pattern recognition and data analysis on the Grassmannian and flag manifold. We also look forward to optimizing the algorithms introduced in this dissertation for more efficient implementation on real-world applications.

The code for implementing the experiments and algorithms in this dissertation will be made publicly available for transparency and reproducibility.

Bibliography

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematicae*, 80(2):199–220, 2004.

[2] Awad H Al-Mohy and Nicholas J Higham. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM Journal on Scientific Computing*, 34(4):C153–C169, 2012.

[3] Awad H Al-Mohy, Nicholas J Higham, and Samuel D Relton. Computing the fréchet derivative of the matrix logarithm and estimating the condition number. *SIAM Journal on Scientific Computing*, 35(4):C394–C410, 2013.

[4] Sherif Azary and Andreas Savakis. Grassmannian sparse representations and motion depth surfaces for 3d action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 492–499, 2013.

[5] C Bach, D Ceglia, L Song, and F Duddeck. Randomized low-rank approximation methods for projection-based model order reduction of large nonlinear dynamical problems. *International Journal for Numerical Methods in Engineering*, 2018.

[6] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *2010 48th Annual allerton conference on communication, control, and computing (Allerton)*, pages 704–711. IEEE, 2010.

[7] J.R. Beveridge, Bruce Draper, Jen-Mei Chang, Michael Kirby, Holger Kley, and Chris Peterson. Principal angles separate subject illumination spaces in YDB and CMU-PIE. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29, 2008.

[8] Åke Björck and Gene H Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973.

[9] Jen-Mei Chang. *Classification on the Grassmannians: theory and applications*. Colorado State University, 2008.

[10] Jen-Mei Chang, J.R. Beveridge, Bruce Draper, Michael Kirby, Holger Kley, and Chris Peterson. Illumination face spaces are idiosyncratic. In *IPCV'06*, volume 2, pages 390–396. CSREA Press, June 2006.

[11] Jen-Mei Chang, Michael Kirby, Holger Kley, Chris Peterson, J. Ross Beveridge, and Bruce Draper. Examples of set-to-set pattern classification. In *Mathematics in Signal Processing Conference Digest*, pages 102–105, Royal Agricultural College, Cirencester, U.K., December 2006. The Insititute for Mathematics and its Applications.

[12] Jen-Mei Chang, Michael Kirby, and Chris Peterson. Set-to-set face recognition under variations in pose and illumination. In *2007 Biometrics Symposium*, Baltimore, MD, September 2007.

[13] Sofya Chepushtanova, Christopher Gittins, and Michael Kirby. Band selection in hyperspectral imagery using sparse support vector machines. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XX*, volume 9088, page 90881F. International Society for Optics and Photonics, 2014.

[14] Sofya Chepushtanova and Michael Kirby. Sparse Grassmannian embeddings for hyperspectral data representation and classification. *IEEE Geoscience and Remote Sensing Letters*, 14(3):434–438, 2017.

[15] Sofya Chepushtanova, Michael Kirby, Chris Peterson, and Lori Ziegelmeier. An application of persistent homology on Grassmann manifolds for the detection of signals in hyperspectral imagery. In *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Milan, Italy, 2015*, 2015.

[16] John H Conway, Ronald H Hardin, and Neil JA Sloane. Packing lines, planes, etc.: Packings in grassmannian spaces. *Experimental mathematics*, 5(2):139–159, 1996.

[17] Jing Dong, Zhichao Xue, Jian Guan, Zi-Fa Han, and Wenwu Wang. Low rank matrix completion using truncated nuclear norm and sparse regularizer. *Signal Processing: Image Communication*, 68:76–87, 2018.

[18] Bruce Draper, Michael Kirby, Justin Marks, Tim Marrinan, and Chris Peterson. A flag representation for finite collections of subspaces of mixed dimensions. *Linear Algebra and its Applications*, 451:15–32, 2014.

[19] Alan Edelman, Tomás A Arias, and Steven T Smith. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

[20] Richard Everson and Lawrence Sirovich. Karhunen–loeve procedure for gappy data. *JOSA A*, 12(8):1657–1664, 1995.

[21] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012.

[22] Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *2011 international conference on computer vision*, pages 999–1006. IEEE, 2011.

[23] Mehrtash T Harandi, Conrad Sanderson, Sareh Shirazi, and Brian C Lovell. Graph embedding discriminant analysis on grassmannian manifolds for improved image set matching. In *CVPR 2011*, pages 2705–2712. IEEE, 2011.

[24] Jun He, Laura Balzano, and Arthur Szlam. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1568–1575. IEEE, 2012.

[25] Yi Hong, Roland Kwitt, Nikhil Singh, Brad Davis, Nuno Vasconcelos, and Marc Niethammer. Geodesic regression on the grassmannian. In *European Conference on Computer Vision*, pages 632–646. Springer, 2014.

[26] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

[27] M. Kirby and C. Peterson. Visualizing data sets on the grassmannian using self-organizing mappings. In *2017 12th International Workshop on Self-Organizing Maps and Learning Vector Quantization, Clustering and Data Visualization (WSOM)*, pages 1–6, June 2017.

[28] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

[29] Teuvo Kohonen. The self-organizing map. *Neurocomputing*, 21(1):1–6, 1998.

[30] Teuvo Kohonen. Essentials of the self-organizing map. *Neural Networks*, 37:52–65, 2013.

[31] Teuvo Kohonen, Erkki Oja, Olli Simula, Ari Visa, and Jari Kangas. Engineering applications of the self-organizing map. *Proceedings of the IEEE*, 84(10):1358–1384, 1996.

[32] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 8:30–37, 2009.

[33] Sriram Kumar and Andreas Savakis. Robust domain adaptation on the l1-grassmannian manifold. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 103–110, 2016.

[34] D Landgrebe. Aviris nw indiana's indian pines 1992 data set, 1992.

[35] Lek-Heng Lim, Ken Sze-Wai Wong, and Ke Ye. Numerical algorithms on the affine grassmannian. *arXiv preprint arXiv:1607.01833*, 2016.

[36] Mengyi Liu, Ruiping Wang, Zhiwu Huang, Shiguang Shan, and Xilin Chen. Partial least squares regression on grassmannian manifold for emotion recognition. In *Proceedings of the 15th ACM on International conference on multimodal interaction*, pages 525–530. ACM, 2013.

[37] Tzu-Yu Liu, Thomas Burke, Lawrence P Park, Christopher W Woods, Aimee K Zaas, Geoffrey S Ginsburg, and Alfred O Hero. An individualized predictor of health and disease using paired reference and target samples. *BMC bioinformatics*, 17(1):47, 2016.

[38] Yasunori Nishimori, Shotaro Akaho, and Mark D Plumbley. Riemannian optimization method on the flag manifold for independent subspace analysis. In *International Conference on Independent Component Analysis and Signal Separation*, pages 295–302. Springer, 2006.

[39] Stephen O'Hara, Kun Wang, Richard A Slayden, Alan R Schenkel, Greg Huber, Corey S O'Hern, Mark D Shattuck, and Michael Kirby. Iterative feature removal yields highly discriminative pathways. *BMC genomics*, 14(1):832, 2013.

[40] Vishal M Patel, Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Visual domain adaptation: A survey of recent advances. *IEEE signal processing magazine*, 32(3):53–69, 2015.

[41] Kathleen H Rubins, Lisa E Hensley, Victoria Wahl-Jensen, Kathleen M Daddario DiCaprio, Howard A Young, Douglas S Reed, Peter B Jahrling, Patrick O Brown, David A Relman, and Thomas W Geisbert. The temporal program of peripheral blood gene expression in the response of nonhuman primates to ebola hemorrhagic fever. *Genome biology*, 8(8):R174, 2007.

[42] Jon M Selig. *Geometric fundamentals of robotics*. Springer Science & Business Media, 2004.

[43] David A Shaw and Rama Chellappa. Regression on manifolds using data-dependent regularization with applications in computer vision. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 6(6):519–528, 2013.

[44] Sima Taheri, Pavan Turaga, and Rama Chellappa. Towards view-invariant expression analysis using analytic shape manifolds. In *Face and Gesture 2011*, pages 306–313. IEEE, 2011.

[45] Pavan Turaga and Rama Chellappa. Locally time-invariant models of human activities using trajectories on the grassmannian. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2435–2441. IEEE, 2009.

[46] Bart Vandereycken. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[47] Kun Wang, Stanley Langevin, Corey S O'Hern, Mark D Shattuck, Serenity Ogle, Adriana Forero, Juliet Morrison, Richard Slayden, Michael G Katze, and Michael Kirby. Anomaly detection in host signaling pathways for the early prognosis of acute infection. *PloS one*, 11(8):e0160919, 2016.

[48] Tiesheng Wang and Pengfei Shi. Kernel grassmannian distances and discriminant analysis for face recognition from image sets. *Pattern Recognition Letters*, 30(13):1161–1165, 2009.

[49] Xinchao Wang, Wei Bian, and Dacheng Tao. Grassmannian regularized structured multi-view embedding for image classification. *IEEE Transactions on Image Processing*, 22(7):2646–2660, 2013.

[50] Ke Ye, Ken Sze-Wai Wong, and Lek-Heng Lim. Optimization on flag manifolds. *arXiv e-prints*, page arXiv:1907.00949, Jul 2019.

[51] DC Youla. A normal form for a matrix under the unitary congruence group. *Canadian Journal of Mathematics*, 13:694–704, 1961.

[52] Ralf Zimmermann, Benjamin Peherstorfer, and Karen Willcox. Geometric subspace updates with applications to online adaptive nonlinear model reduction. *SIAM Journal on Matrix Analysis and Applications*, 39(1):234–261, 2018.